Convolutional SVM Networks for Object Detection in UAV Imagery

Yakoub Bazi, *Senior Member, IEEE*, and Farid Melgani, *Fellow, IEEE*

Abstract— **Nowadays, Unmanned Aerial Vehicles (UAV) are viewed as effective acquisitions platforms for several civilian applications. They can acquire images with an extremely high-level of spatial details compared to standard remote sensing platforms. However, these images are highly affected by illumination, rotation and scale changes, which increases further the complexity of analysis compared to those obtained by standard remote sensing platforms. In this paper, we introduce a novel Convolutional Support Vector Machine (CSVM) network for the analysis of this type of imagery. Basically, the CSVM network is based on several alternating convolution and reduction layers ended by a linear SVM classification layer. The convolution layers in CSVM rely on a set of linear SVMs as filter banks for feature map generation. During the learning phase, the weights of the SVM filters are computed through a forward supervised learning strategy unlike the backpropagation algorithm widely used in standard convolutional Neural Networks (CNNs). This makes the proposed CSVM particularly suited to detection problems characterized by very limited training sample availability. The experiments carried out on two UAV datasets related to vehicles and solar panel detection issues, with a 2-centimeter resolution, confirm the promising capability of the proposed CSVM network compared to recent state-of-the-art solutions based on pretrained CNNs.**

Index Terms— **Convolutional SVM filtering, convolutional SVM network, extremely high-spatial resolution images, object detection, supervised feature generation, UAV platforms.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are recognized as very effective systems for collecting images from a very low altitude. The high flexibility of these small, ecologic and silent aerial platforms permits immediate intervention and interactive measurements according to customer's specific needs. Additionally, they allow mapping and monitoring of small areas at extremely fine scales and enables multitemporal acquisitions over the same area at predefined and desired times. These attractive proprieties render them a valid alternative or a complementary solution to satellite sensors particularly for small coverage or inaccessible areas. In the beginning, UAVs were used exclusively for military applications, but due to advancement in technologies and reduction in prices, they became a practical solution for many civilian applications such as precision farming [1]–[3], anomaly detection in archeological sites [4] , and monitoring of urban areas [5], [6].

UAVs can acquire images with extremely high-level of spatial details compared to standard remote acquisition systems such as satellites or airborne. They are very appropriate for detecting specific class of objects, which opens the door to the development of various interesting recognition applications [7]–[12]. Even though promising results have been obtained in these recent developments, they mainly rely on handcrafted-feature representations. These types of representations would limit the performances of the recognition system as they work well under restricted conditions. Actually, the increase in spatial resolutions (order of few centimeters) poses new challenges for automatic classification as the objects belonging to the same class will simply look very different. Additionally, UAV images are highly affected by illumination, rotation and scale changes which increases further the complexity of identifying robust visual handcrafted features for representing the image content.

Recently deep convolutional neural networks (CNNs) have achieved impressive results on a variety of applications including image classification [13]–[16], object detection [17]–[20], and image segmentation [21], [22]. Thanks to their sophisticated structure, they have the ability to learn powerful generic image representations in a hierarchical way compared to state-of-the-art shallow methods based on handcrafted features. Modern CNNs are made up of several alternating convolution and pooling layers followed by some fully connected layers. The feature maps produced by the convolution layers are usually fed to a nonlinear gating function such as the Rectified Linear Unit (ReLU). Then the output of this activation function can further be subjected to normalization (i.e., local response normalization). The whole CNN architecture is trained end-to-end using the backpropagation algorithm with dropout regularization [23] to reduce overfitting. It is worth recalling that recent deeper CNNs (winner of the ImageNet Large-Scale Visual

This work was supported by the Deanship of Scientific Research at King Saud University through the Local Research Group Program under Project RG-1435-055.

Yakoub Bazi, is with the Computer Engineering department, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia (e-mail: ybazi@ksu.edu.sa. Farid Melgani is with the department of Information Engineering and Computer Science, University of Trento, Via Sommarive 9, I-38123, Trento, Italy E-mail: melgani@disi.unitn.it.

Recognition ILSVRC14 and ILSVRC15 challenges) use inception modules [15] and residual learning [14].

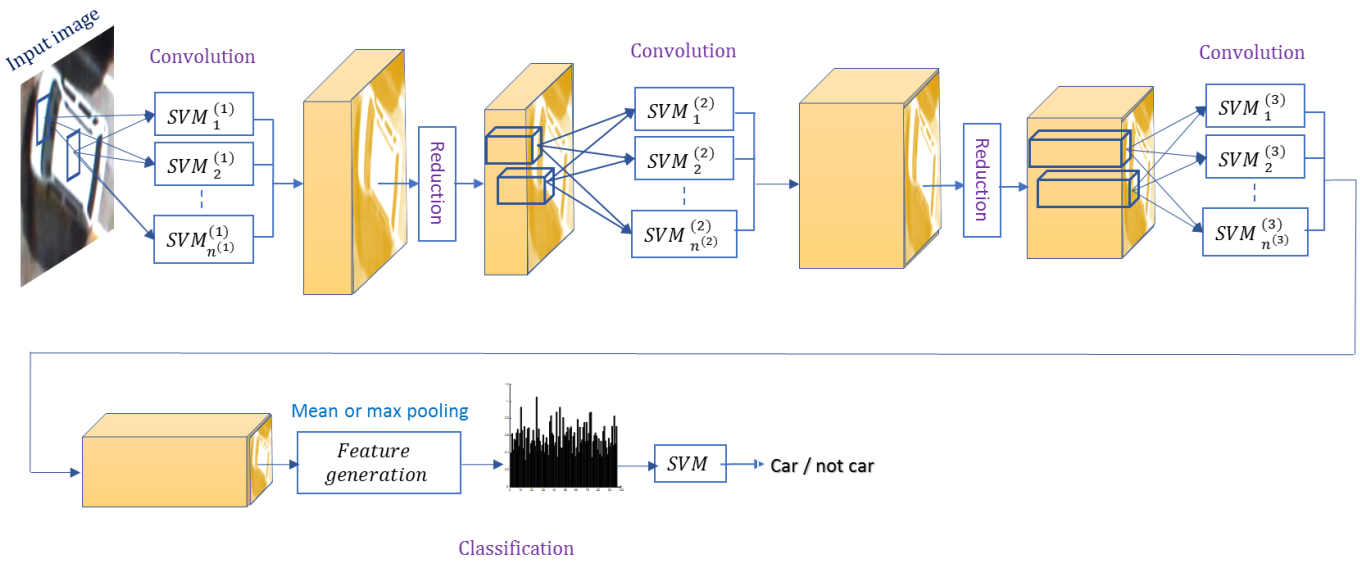


Figure 1. Example of a CSVM network with: three convolution layers; two reduction layers, and a classification layer placed on the top.

Usually, CNNs perform well for analyzing datasets with large labelled data. Yet they are prone to overfitting when dealing with datasets with limited labeled data. For these scenarios, it is has been shown that it is more appealing to transfer knowledge from CNNs (such as AlexNet [16], VGG-VD [24], GoogLeNet [15], and ResNet [14]) pretrained on an auxiliary recognition task with very large labeled data instead of training a CNN from scratch [25]–[28]. While the possible transfer learning solutions include fine-tuning the pretrained CNN on the labeled data of the target dataset or to exploit the CNN feature representations with an external classifier. We refer the reader to [26] where the authors introduce and investigate several factors affecting the transferability of these representations.

In the remote sensing context, some contributions have used recently these strategies for handling small datasets acquired either by airborne or satellite sensors mainly composed of images with spatial resolution up to 30 cm. For instance, in [29] the authors used a pretrained CNN to generate an initial feature representation of the remote sensing images under analysis. After this step, they reshaped the outputs of the two last fully connected layers into 2D arrays and trained a new CNN composed of two convolutions and two fully connected layers followed by a softmax classifier. As pretrained CNN they used the Overfeat model [30], which is an improved version of the AlexNet model [16]. By contrast Esam *et al.*[31] used the output of the last fully connected layer of the CNN model proposed by Chatfield *et al.* [25] as input to a sparse autoencoder for learning a new feature representation. Then they tailored the sparse autoencoder to view the classification problem from discriminative and reconstruction perspectives. Finally, Fan *et al.* [32] investigated the problem of transferring features from different CNN models trained also on very large auxiliary labeled dataset. They considered two different approaches for carrying out feature extraction. For the first approach, they used the last fully connected layer, while for the second one they extracted dense features from the convolution layer at multiple scales and then used different encoding techniques to generate the final representation of the image. For both solutions, they used a support vector machine (SVM) classifier for training [33] .

As mentioned previously, UAV images add more challenges compared to standard remote sensing images, as they are characterized by improved spatial resolutions; large intra-class variation; significant differences in image statistics in addition to variation in illumination, viewpoint, rotation, and scale changes. Thus we expect that a simple fine-tuning of the network or feeding the feature representations to an external classifier will not easily address this problem, which calls for further research developments.

In this paper, we propose an alternative learning strategy based on SVMs for handling these scenarios. SVMs are among the most popular supervised classifiers available in the literature. They rely on the margin maximization principle which makes them less sensitive to overfitting problems. They have been intensively used in conjunction with handcrafted features for solving various recognition problems. In addition, as discussed previously, they are also commonly placed on the top of a CNN feature extractor for carrying out the classification task [26]. In this work, we extend them to act as convolutional filters for the supervised generation of features maps. We term this network as convolutional SVM (CSVM). Compared to standard CNNs, the CSVM introduces a new convolution trick based on SVMs, and does not rely on the backpropagation algorithm for training.

Basically, the CSVM network is based of several alternating convolution and reduction layers followed by a classification layer (Figure 1). Each convolution layer uses a set of linear SVMs as filter banks, which are convolved with the feature maps produced by the precedent layer to generate a new set of features maps. For the first convolution layer, the SVM filters are convolved with the original input images. The SVM weights of each convolution layer are computed directly in a supervised way by training on patches (extracted from the

previous layer) representing the objects of interest. The feature maps produced by the convolution layers are then fed to a nonlinear gating function such as ReLU. The reduction layer in CSVM works in

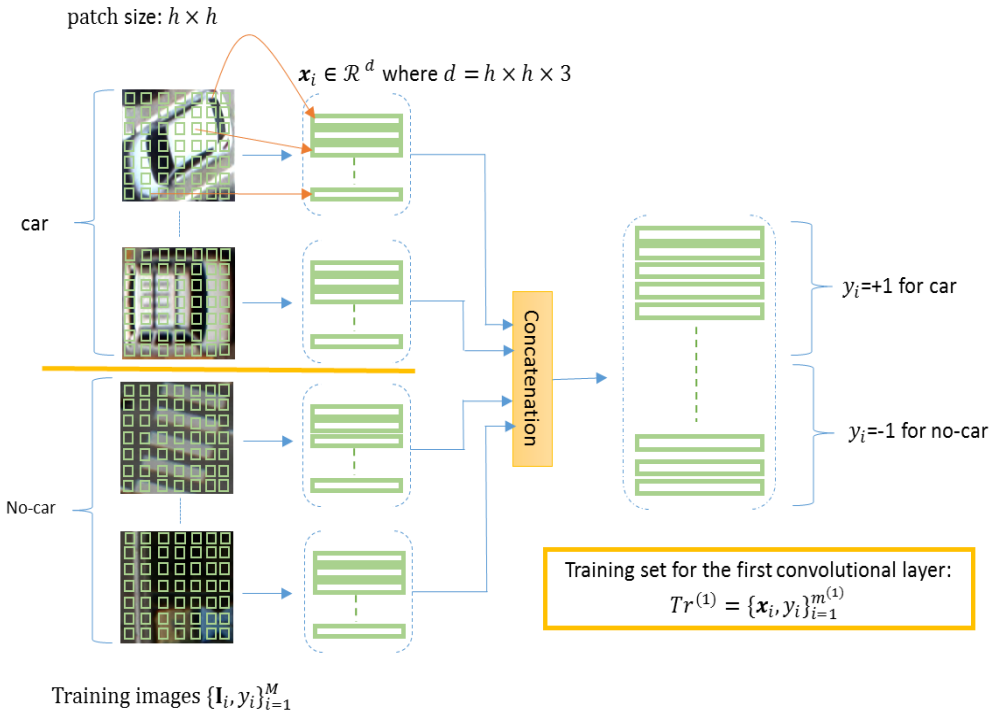


Figure 2. Training set generation for the first convolutional layer.

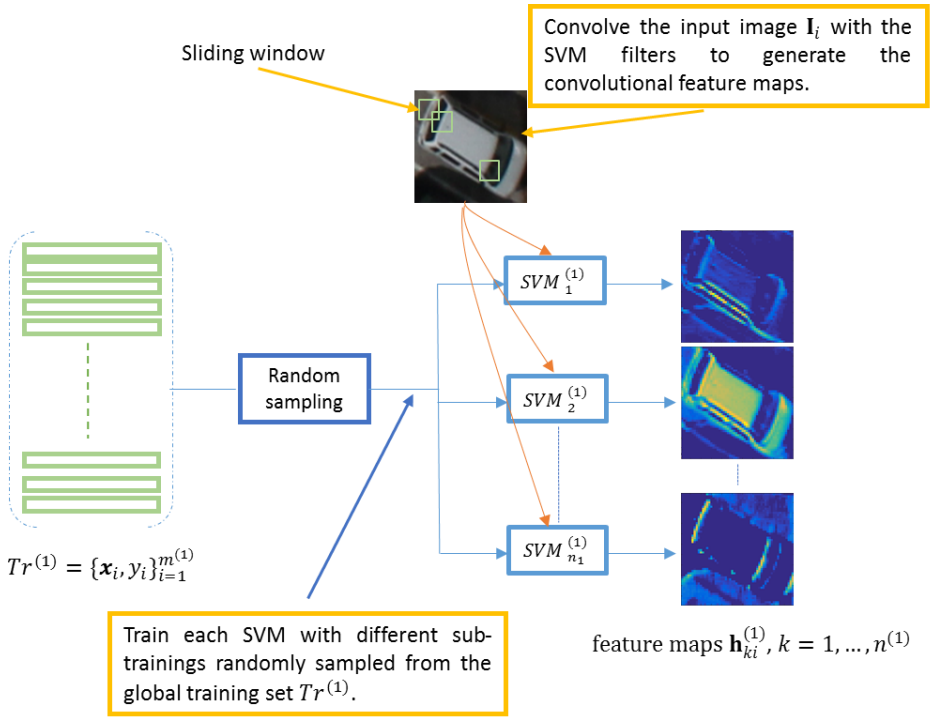


Figure 3. Feature maps obtained by the first SVM convolutional layer.

a similar way to the pooling layer in CNN. It takes small rectangular blocks from the convolutional layer and subsamples it to produce a single output from each block. Finally, the high level representations obtained by the network are fed again to a linear SVM classifier for carrying out the classification task.

The major contribution of this work can be summarized as

follows: (1) Introduce a novel CSVM network suitable for problems with small training data. (2) Extend SVMs to act as convolutional filters for generating feature representation maps. (3) Present a forward supervised learning strategy for computing the weights of the network. (4) In the experiments, we validate the method on UAV images with 2-centimeter resolution acquired over urban areas related to vehicles and solar panels detections.

The rest of the paper is organized as follows. Section II provides a general description of the proposed CSVM network. Section III presents the results on two UAV datasets. Finally, Section IV presents the conclusions of the work and future developments.

II. CSVM Network Description

A. SVM Convolution layer

As mentioned previously, the SVM convolution layer is the core building of the CSVM network. Specifically this layer uses linear SVMs as convolutional filters for generating the feature maps. Furthermore, the weights of these filters are learned using a forward supervised learning method unlike the weights of conventional CNNs, which are computed via backpropagation. In the following, we detail this method for the first convolution layer. The generalization to the next convolution layers is straightforward.

1) *Construction of the Training set*: If we let $\{\mathbf{I}_i, y_i\}_{i=1}^{M}$ be the training set composed of M positive and negative RGB images with $\mathbf{I}_i \in \boldsymbol{\mathcal{R}}^{\boldsymbol{r\times c\times 3}}$ where r and c refer the number of rows and columns of the image, and $y_i \in \{+1, -1\}$ denotes the class label. The positive images contains the object of interest, whereas the negatives ones represent background. From each image $\mathbf{I}_i$, we extract a set of patches of size $h \times h \times 3$ (i.e., three channels) and represent them as feature vectors $\mathbf{x}_i$ of dimension d, where $d = h \times h \times 3$. After processing all images, we obtain a global training set $Tr^{(1)} = \{\mathbf{x}_i, y_i\}_{i=1}^{m^{(1)}}$ of size $m^{(1)}$ as shown in Figure 2.

2) *Training the SVM filter banks:* In this step, we learn a set of SVM filters on different sub-training sets $Tr_{sub}^{(1)} = \{\mathbf{x}_i, y_i\}_{i=1}^{l}$ of size l randomly sampled from the global training set $Tr^{(1)}$. The weight vector $\boldsymbol{w} \in \boldsymbol{\mathcal{R}}^{\boldsymbol{d}}$ and bias $b \in \boldsymbol{\mathcal{R}}$ of each SVM filter are determined by optimizing the following unconstrained optimization problem [34], [35]:

$$\min_{\boldsymbol{w},b} \boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i) \quad (1)$$

where C is a penalty parameter.

A common choice for the loss functions $\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i)$ is the hinge loss $\max(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b), 0)$ and the squared hinge loss $\max(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b), 0)^2$ referring to L1-SVM and L2-SVM, respectively. In this work, we shall consider L1-SVM, though the L2-SVM formulation could be considered as well. For simplicity, we omit the bias term and represent the weights of the SVM filters obtained after training as $\left\{\mathbf{w}_k^{(1)}\right\}_{k=1}^{\boldsymbol{n^{(1)}}}$, where $\mathbf{w}_k^{(1)} \in \boldsymbol{\mathcal{R}}^{h\times h\times 3}$ refers to *k*th-SVM filter weight matrix, while $n^{(1)}$ is the number of filters. Then, the complete weights of this convolution layer could be grouped into filter bank of four dimensions $\boldsymbol{W^{(1)}} \in \boldsymbol{\mathcal{R}}^{h\times h\times 3\times n^{(1)}}$.

3) *Generation of the convolutional feature maps:* In this step, we convolve each training image $\{\mathbf{I}_i\}_{i=1}^{M}$, with the SVM filters to generate a set of 3D hyper-feature maps $\{\mathbf{H}_i^{(1)}\}_{i=1}^{M}$. Here $\mathbf{H}_i^{(1)} \in \boldsymbol{\mathcal{R}}^{r^{(1)}\times c^{(1)}\times n^{(1)}}$ is the new feature representation of image $\mathbf{I}_i$ composed of $n^{(1)}$ feature maps (Figure 3). To obtain the *k*th feature map $\mathbf{h}_{ki}^{(1)}$, we convolve the *k*th SVM filter with a set of sliding windows of size $h \times h \times 3$ (with a predefined stride) over the training image $\mathbf{I}_i$ (Figure 4):

$$\mathbf{h}_{ki}^{(1)} = f\left(\mathbf{I}_i * \mathbf{w}_k^{(1)}\right), k = 1, \ldots, n^{(1)} \quad (2)$$

where $*$ is the convolution operator and f is the ReLU activation function.

In the following algorithm, we present a practical implementation of this convolutional layer. For more clarity, we use Matlab notations.

Algorithm: *SVM convolutional layer*

Input: - Training images $\{\mathbf{I}_i, y_i\}_{i=1}^{M}$
- Number of SVM filters: $n^{(1)}$
- Filter parameters (width h, and *stride*)
- Number of Training patches: l

Output: Feature cube: $\mathbf{H}^{(1)} \in \mathcal{R}^{r^{(1)}\times c^{(1)}\times n^{(1)}\times M}$

Start:

1: $Tr^{(1)} = global\_training(\{\mathbf{I}_i, y_i\}_{i=1}^{M}, h)$; // For example use im2col of Matlab to extract the patches.

2: $\boldsymbol{W^{(1)}} = learn\_SVM\_filters\left(Tr^{(1)}, n^{(1)}, l\right)$ // Train $n^{(1)}$-
//SVMs on different sub-trainings $Tr_{sub}^{(1)}$ of size l randomly
//sampled from $Tr^{(1)}$. Use Libilinear software package for
// SVM training [34].

3: $\mathbf{H}^{(1)} = convolution\left(\{\mathbf{I}_i\}_{i=1}^{M}, \boldsymbol{W}^{(1)}\right)$; // use *vl_nnvonv* of MatConvNet [36]

4: $\mathbf{H}^{(1)} = ReLU\left(\mathbf{H}^{(1)}\right)$

End

It is worth recalling that this algorithm is also valid for the next convolutional layers except that actual inputs are the hyper-feature maps produced by the previous layer as shown in Figure 1.

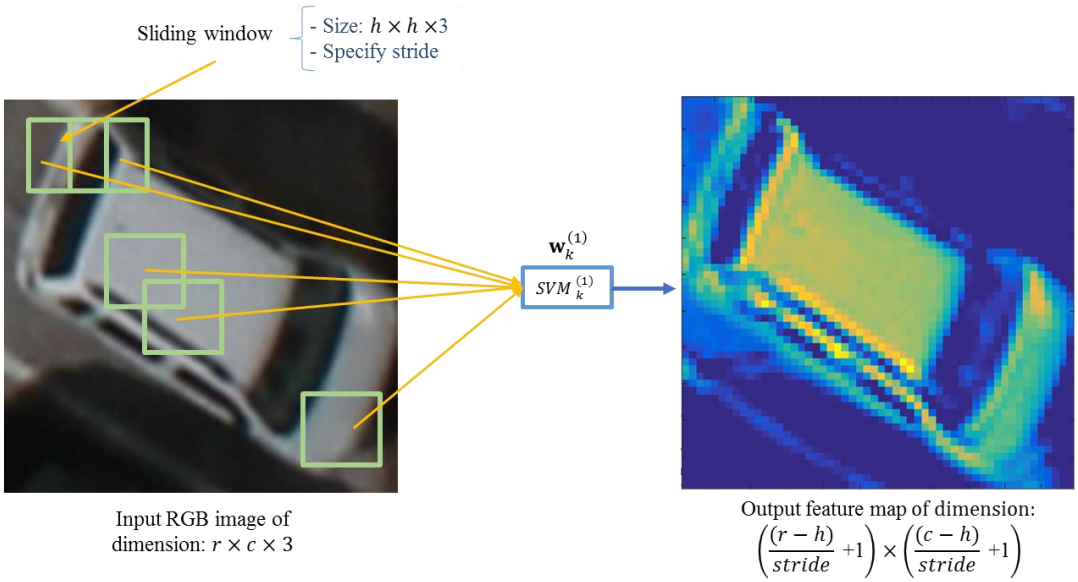


Figure 4. Example of a feature map produced by the *k*th SVM convolutional filter.

B. Reduction Layer

Here the spatial reduction layer acts in a similar way to the spatial pooling layer in CNNs. It is commonly inserted between two successive convolution layers to reduce the spatial size of the representation by selecting the most useful features for the next layers. Basically, it takes small rectangular blocks from the convolutional features maps and subsamples them to produce a single output from each block. There are several ways to perform spatial reduction, such as taking the average or the maximum, or a learned linear combination of the values in the block. In this work, we use the max pooling operator for carrying out reduction. As for CNNs, the output of the reduction layer could be followed by a local response normalization (LNR) layer. However, we experimentally found that this layer does not affect much the results on the two UAV datasets used in the experiments.

C. Feature generation and classification

After several SVM convolutional and reduction layers, the high level reasoning of the network is done by training a binary SVM classifier on the high level features extracted from the last layer (Figure 1). If we let $\{\mathbf{H}_i^{(L)}, y_i\}_{i=1}^M$ be the hyper-feature maps obtained by the last computing layer L (convolution or reduction depending on the network architecture). If we suppose also that each hyper-feature map $\mathbf{H}_i^{(L)}$ is composed of $n^{(L)}$ feature maps. Then a possible solution of extracting the high level feature vector $\boldsymbol{z}_i \in \boldsymbol{\mathcal{R}}^{\boldsymbol{n}^{(L)}}$ of dimension $n^{(L)}$ for the training image $\mathbf{I}_i$ could be simply done computing the mean or max value for each feature map. We refers to these two types of feature generation in the experiments as mean and max spatial pooling, respectively.

III. EXPERIMENTS

A. Dataset Description

In the experiments, we assess the proposed CSVM network on two UAV datasets related to vehicles and solar panels, respectively. The images composing these two datasets were acquired at different times over different areas in or near the city of Trento, Italy, by means of a UAV equipped with imaging sensors spanning the visible range (Figure 5). All acquisitions were performed with a picture camera Canon EOS 550D characterized by a CMOS APS-C sensor with 18 megapixels. The obtained RGB images have a spatial resolution of approximately 2 cm and 8 bit of radiometric resolution. For both datasets, we resize all images to the dimension 224×224 pixels commonly used by CNNs.

1) *Car dataset*: The determination of the number of vehicles on roads or in parking lots represents one of the most discussed and interesting issues in the field of object detection in remotely sensed imagery. It opens the way to solve urban problems that could be encountered almost every day. Especially in big cities, knowing the concentration of cars in roads or in parking lots can be extremely interesting for local administrations in order to optimize the urban traffic planning and management. This dataset is composed of 200 training images (100 car and 100 background) acquired over two parking lots inside the city of Trento. On the other side, the test set is composed of 1000 images (500 cars and 500 background) and it was acquired over three parking lots near the city. All acquisitions were made at completely different times and under different acquisition conditions. Figure 6 shows sample of images showing individual cars or group of cars parked with different orientations.

Figure 5. UAV used for the acquisition of the images.

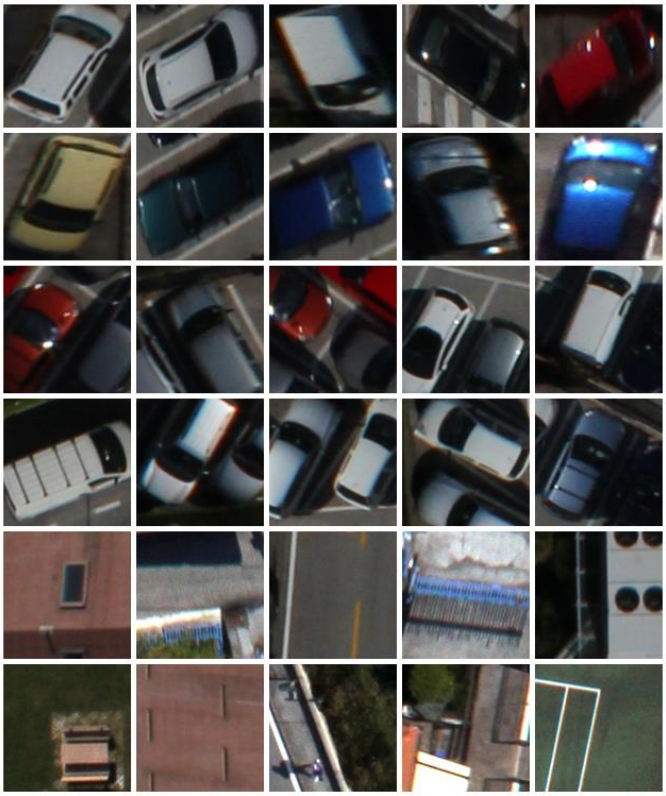


Figure 6. Car dataset: Example of cars (First fourth rows) and backgound (last two rows) acquired over different parking lots over the city of Trento.

2) *Solar-Panel dataset:* Solar panels are gradually invading urban areas to produce clean energy as they do not give damaging emissions compared to traditional sources of energy. The UAV technology represents an appropriate solution to monitor these new types of objects. For this dataset also, several acquisitions have been performed to simulate different scenarios (Figure 7). As for car dataset, the training set is composed of 200 images, whereas the test is composed of 1000 images.

Figure 7. Solar panel dataset: Example of solar panels (first up to foutrh rows) and background (last two rows) acquired over the city of Trento.

B. Experiment Setup

First, we present the preliminary results using small, medium and large CSVM networks. Figure 8 details the architecture of these networks including the parameters of the convolution layer in addition to the reduction and classification layers. Specifically, we propose a configuration with five convolution layers, three reduction layers and a classification layer. For training each SVM filter in the convolution layers, we extract randomly l=200 patches (100 for object and 100 for background) from the training images as shown in the methodological part. For an efficient computation of the weights of the SVM filters we use the multi-core Liblinear software package [34]. The estimation of the penalty parameter C for each SVM is done via a three-fold cross-validation procedure in the range $[10^{-1}, 10^{3}]$. Additionally as usually done for CNNs, we preprocess all images by subtracting the average image of all training images before feeding them to the network.

In the second experiment, we carry out a sensitivity analysis to study the effects of the filter sizes on the performances of the network. Finally, in the last experiment, we compare our results to state-of-the art methods based on pretrained CNNs.

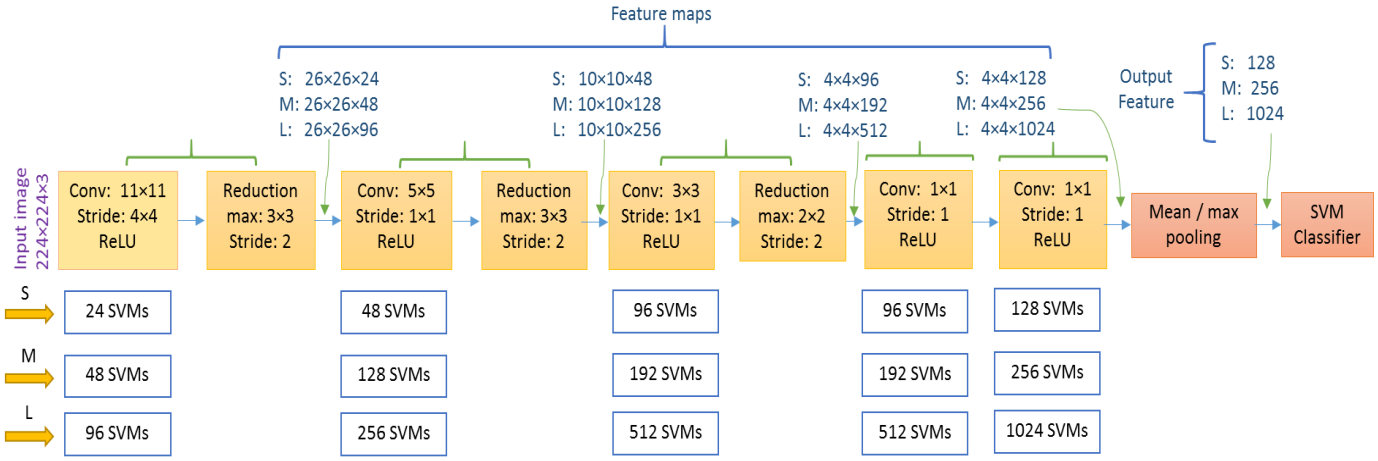


Figure 8. CSVM network: Small (S), Medium (M) and Large (L).

For performance evaluation, we present the results in terms of average accuracy (i.e., $AA = \frac{UA+PA}{2}$) where PA is the producer's accuracy corresponding to error of omission (i.e., $PA = \frac{TP}{N_{obj}}$) and UA is user's accuracy corresponding to the error of commission (i.e., $UA = \frac{TP}{TP+FP}$). TP is true positives, number of car/solar-panel images correctly identified, FP is false positives, number of car/solar-panel images incorrectly identified, and N_{obj} is the total number of car/solar-panel images. All experiments were carried out on a laptop having a 2.50 GHz Intel(R) Core(TM) i7 processor and a memory RAM of 16 GB.

C. Results

1) *Preliminary results*: Figure 9 provides an illustration of the feature maps produced by the different convolution layers of two small networks trained on car and solar panel datasets, respectively. As can be seen, the SVM filters are able to provide different feature maps representation of the solar/panel test image. In particular some of these filters yield high responses clearly highlighting the presence of the objects of interest. In Figures 10 and 11, we report the classification accuracies obtained by the small, medium and large networks for both datasets, respectively. Besides the network output, these figures show also the intermediate results obtained after each convolutional layer.

In general, all these three configurations provide competing results. However, by taking into account the computation time and the number of weights for each configuration (Table 1), the small network appears to be more appropriate. While the medium and large networks uses a large number of filters, they yield in some cases reduced accuracies. This could be explained by the increase in feature dimensions, which needs the definition of particular training strategy dealing with the resulting curse of dimensionality, which is beyond the scope of this work. In the rest of the paper, we shall focus on the small network architecture.

By averaging results obtained for the two datasets, we notice that the small network reaches stable results after the second convolutional layer with mean pooling, whereas it requires four convolutional layers for max pooling. In detail using mean pooling, we obtain average accuracies of (92.56%, 96.03% 95.96%, 96.36% and 96.01%) after each convolution layer, whereas for max pooling we obtain accuracies of (89.12%, 92.49, 94.27%, 95.23% and 95.11%). These preliminary results, *a-priori* suggest that for these two datasets, at least two convolutional layers are needed to generate good representative features maps of the object of interest. In addition, increasing the number of layers can improve the results particularly for solar-panel dataset. For car dataset, we observe some variations in the accuracy (but less than 1%) when mean pooling is used.

2) *Sensitivity analysis:* In this experiment, we analyze further the small network architecture by considering different parameters for the convolution filters. To this end, we use different filter sizes while we keep their number per-layer fixed as before. In addition, we set the strides in each convolution layer to the new values $(2, 1, 1, 1)$, while they were fixed in the previous experiment to $(4, 1, 1, 1)$. This will allow us to obtain feature maps with large spatial sizes. For feature generation we consider only mean pooling as it yielded better results compared to max pooling. Although all configurations yielded competing results as shown in Figure 12, the one using the filters $(7 \times 7, 1 \times 1, 3 \times 3, 1 \times 1, 1 \times 1)$ looks more stable for both datasets. For car dataset, it yields an AA of 93.74% and 96.1% for the first and second layers and ends with an accuracy 97% at the network output. This corresponds to an increase of 1.22% compared to the initial configuration proposed in the first experiment with the following filters $(11 \times 11, 5 \times 5, 3 \times 3, 1 \times 1, 1 \times 1)$.

Regarding solar panel dataset, the first layer leads to an AA of 89.3% and the second to 95.59%, and ends with an accuracy of 96.97% at the network output. Here again, we observe an increase of 0.93% compared to the initial configuration. It is worth to note that the network corresponding to this configuration uses less SVM weights (6.80E04), while the computation time has increased to 7 minutes. This increase is due to the extra convolution operations done on feature maps with increased spatial sizes (54×54×24 up to 11×11×128 pixels from the first layer to last layer, respectively).

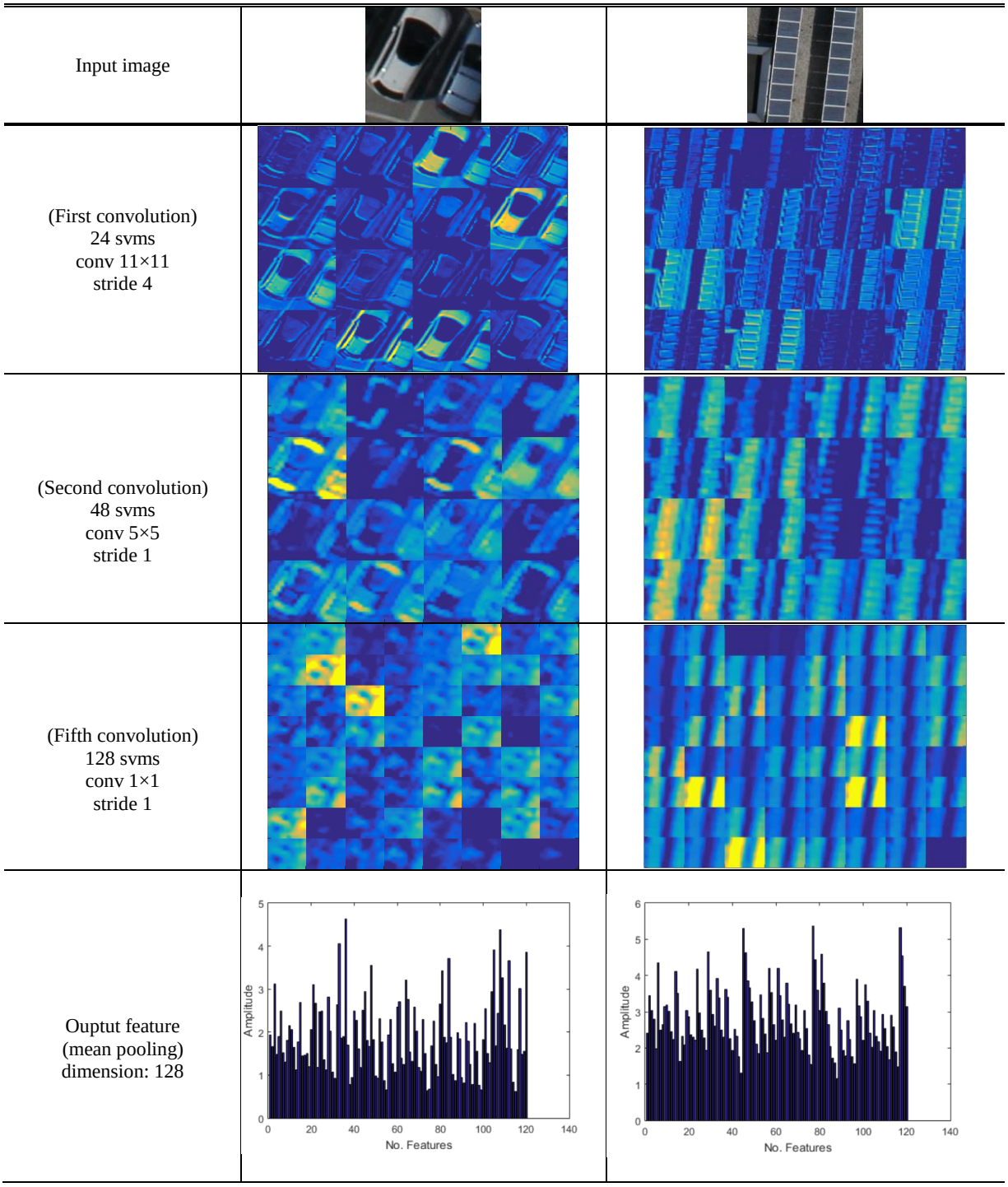
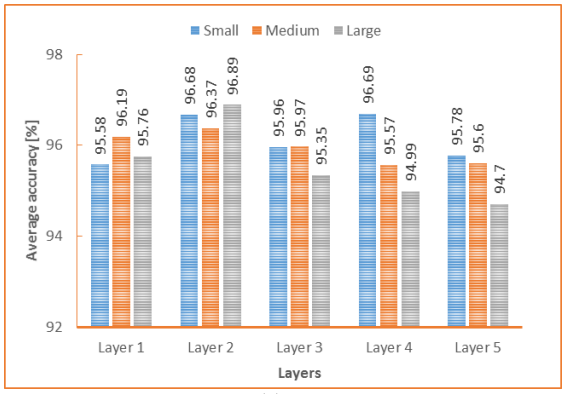
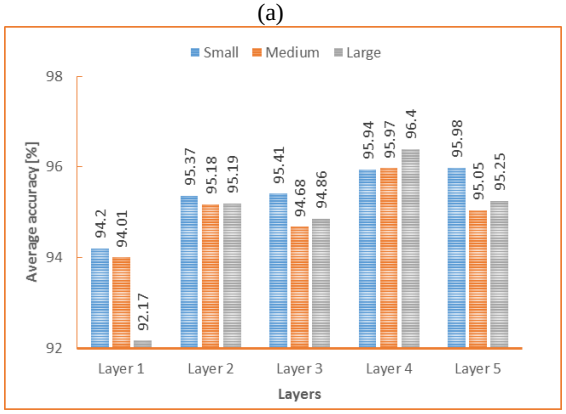


Figure 9. Output feature maps for car/solar-panel test images (produced by the first, second and last convolutional layers). The last row represents the output feature representation obtained by applying mean pooling to the last convolutional layer.

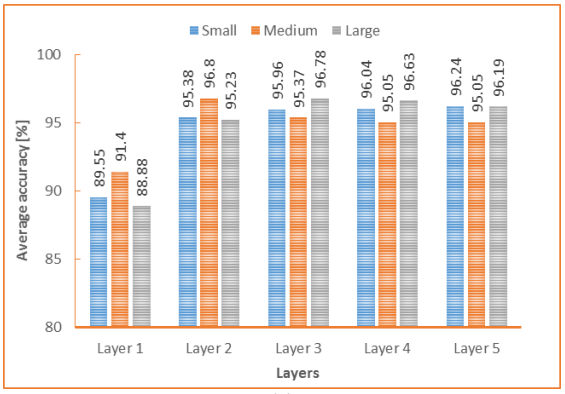


(a)

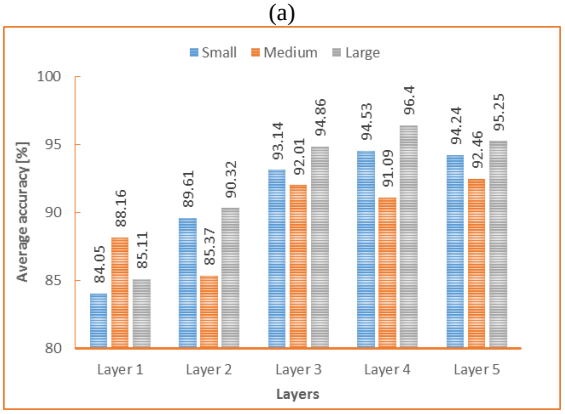


(b)

Figure 10. Average classification accuracy obtained by small, medium and large CSVMs on car dataset. For each layer, we use a simple (a) mean and (b) max pooling for generating the output features fed to the SVM classifier.



(a)



(b)

Figure 11. Average classification accuracy obtained by small, medium and large CSVMs solar-panel dataset. For each layer, we use a simple (a) mean and (b) max pooling for generating the features fed to the SVM classifier.

3) *Comparison with pretrained CNNs:* In this experiment, we compare our results to the state-of-the-art solutions based on pretrained CNNs. Specifically, we feed the UAV images to CNNs (pretrained on a very large labeled dataset such as ImageNet [37]) to generate the corresponding feature representations and then use linear SVM for training and classification. In the experiments, we use different pretrained CNNs from MatConvNet [36] including AlexNet [16], VGGM [25], VGG-16 [24], GoogLeNet [15], and ResNet [13]. The AlexNet network was the winner of the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge). It is composed of 5 convolutional layers, 3 pooling layers, and 3 fully-connected layers. The VGGM network has the same number of layers as AlexNet but with different convolution filter sizes. The VGG16 network has a deeper architecture as it is composed of 13 convolutional layers, 5 pooling layers, and 3 fully-connected layers.

The GoogLeNet network introduces the idea of inception module that acts as multiple convolution filters applied to the same input. This network is composed of 9 inception modules and replaces the fully-connected layers with average pooling. The ResNet network is a more advanced architecture which reformulates the layers as learning residual function, with reference to the layers input. Here we use the ResNet50 model, which is composed of 50 layers. It is worth recalling that the features obtained from the first three networks are of dimensions 4096 whereas they are equal to 1000 when using GoogLeNet and ResNet.

Table 2 presents the results using different training set sizes by sampling randomly from the original training set. For CSVM, we use the configuration based on the following filters $(11 \times 11, 5 \times 5, 3 \times 3, 1 \times 1, 1 \times 1)$ and $(7 \times 7, 1 \times 1, 3 \times 3, 1 \times 1, 1 \times 1)$. The results obtained for five trials (different training images) show an interesting behavior of the proposed network.

TABLE I.
NUMBER OF CSVM WEIGHTS FOR SMALL, MEDIUM AND LARGE CONFIGURATIONS.

		Small		**Medium**		**Large**	
Layers	Filter size	Feature size	Weights	Feature size	Weights	Feature size	Weights
Convolution	11×11	363	8736	363	17472	363	34944
	5×5	600	28848	1200	153728	2400	614656
	3×3	432	41568	1152	221376	2304	1180160
	1×1	96	9312	192	37056	512	262656
	1×1	96	12416	192	49408	512	525312
Classification	1×1	128	129	256	257	1024	1025
Total weights			1.01E+05		4.79E+05		2.62E+06
Computation time (Minutes)			3.67		7.37		31.63

TABLE II. COMPARISON OF CSVM AGAINST SEVERAL PRETRAINED CNNS: (A) CAR AND (B) SOLAR PANEL DATASETS. THE RESULTS ARE EXPRESSED IN TERMS OF AVERAGE ACCURACY IN (%) AND STANDARD DEVIATION OVER FIVE TRIALS. TRAIN TIMES ARE RELATED TO THE SCENARIO WITH 100 TRAINING IMAGES (PER CLASS). TEST TIMES ARE COMPUTED FOR 1000 TEST IMAGES.

(A)

Training Image/Class	**ResNet-50**	**ResNet-152**	**GoogLeNet**	**VGG16**	**VGGM**	**AlexNet**	**CSVM** (11×11,5×5, 3×3,1×1,1×1)	**CSVM** (7×7,1×1, 3×3,1×1,1×1)
25	67.07±2.6	64.97±2.1	87.04±1.1	91.98±3.3	86.63±1.4	75.04±5.7	93.86±2.28	**94.26±2.1**
50	82.91±0.8	81.81±1.4	88.47±1.3	92.56±2.2	87.15±5.5	81.10±6.5	93.22±0.62	**95.13±0.2**
100	91.10	90.05	94.74	95.66	88.34	91.93	95.78	**97**
Train/Test Time (Seconds)	75 / 274	270 / 1220	126 / 510	235 / 976	45 / 212	42 / 161	130 / 9	185 / 12

(B)

Training Image/Class	**ResNet-50**	**ResNet-152**	**GoogLeNet**	**VGG16**	**VGGM**	**AlexNet**	**CSVM** (11×11,5×5, 3×3,1×1,1×1)	**CSVM** (7×7,1×1, 3×3,1×1,1×1)
25	54.85±6.7	51.20±0.9	69.64±11.5	71.15±12.2	62.55±16.6	67.20±10.4	91.46±6.08	**94.77±1.37**
50	58.71±11.4	52.65±2.6	72.04±18.3	75.85±15.3	64.45±19.4	69.30±16.2	93.32±3.74	**96.19±1.9**
100	66.43	66.37	84.45	93.59	95.45	93.01	96.24	**96.97**
Train/Test Time (Seconds)	81 / 396	289 / 1300	102 / 509	201 / 988	45 / 213	40 / 154	120 / 10	156 / 12

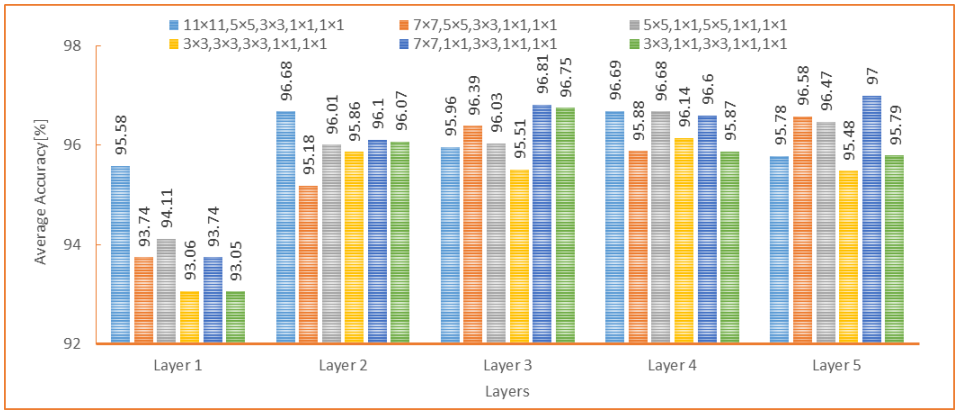
TABLE III. COMPARISON OF CSVM AGAINST HANDCRAFTED FEATURES: (A) CAR AND (B) SOLAR PANEL DATASETS. THE RESULTS ARE EXPRESSED IN TERMS OF AVERAGE ACCURACY (AA), AND TRAIN AND TEST TIMES. 100 TRAINING IMAGES PER CLASS WERE USED. TEST TIMES ARE COMPUTED FOR 1000 TEST IMAGES.

(A)

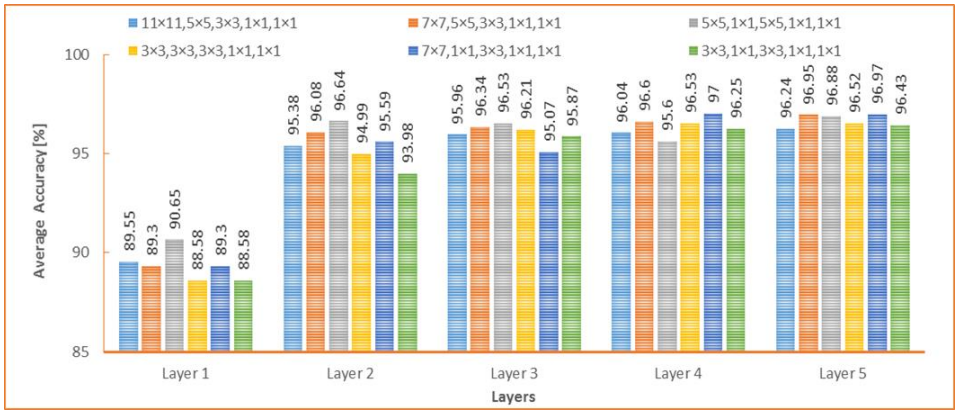
Method	**HOG**	**H-Gradients**	**LBP**	**BOW**	**CSVM** (11×11,5×5, 3×3,1×1,1×1)	**CSVM** (7×7,1×1, 3×3,1×1,1×1)
AA (%)	67.44	75.59	79.39	72.72	95.78	**97**
Train/Test Time (Seconds)	16 / 30	23 / 50	11 / 15	237 / 254	130 / 9	185 / 12

(B)

Method	**HOG**	**H-Gradients**	**LBP**	**BOW**	**CSVM** (11×11,5×5, 3×3,1×1,1×1)	**CSVM** (7×7,1×1, 3×3,1×1,1×1)
AA (%)	85.73	50.27	59.96	75.87	96.24	**96.97**
Train/Test Time (Seconds)	46 / 84	47 / 107	10 / 15	225 / 213	120 / 10	156 / 12



(a)



(b)

Figure 12. Results obtained using different filter sizes for: (a) car and (b) solar-panel datasets. The following strides are used for the five convolution layers: (2, 1, 1, 1, 1). The filters (11×11, 5×5, 3×3, 1×1, 1×1) represent the initial configuration proposed for small CSVM.

It provides better results for all scenarios particularly for reduced training set sizes compared to the solutions based on pretrained CNNs. For example for car dataset, the CSVMs using the mentioned filters, yield accuracies of (95.78%, 93.22% and 93.98%) and (95.78%, 93.22%, and 93.86%) for the scenarios (100, 50, and 25 training images). The closest pretrained CNN is the one based on VGG-16, which provides accuracies (95.66%, 92.56% and 91.98%) while the other networks perform poorly. These promising results are also confirmed for solar panel dataset where the CSVM network exhibits clearly a better behavior for all three cases.

For further analysis, we have compared the results of CSVM to some popular handcrafted feature methods such as: 1) histogram of oriented gradients (HOG) [39], which describe an image based on its local gradients; 2) local binary patterns (LBP) [40], which are simple yet efficient binary descriptors for coding pixel-wise information in textured images; 3) bag-of-visual-words (BOVW) [41], which aim to represent an image by means of a codebook generated from the training images. The occurrence of each visual word is observed in each target image to produce an image signature consisting of the frequencies of all the codebook elements. Finally, high-order gradients (H-gradients) [12], which combine image gradient features at different orders, have also been considered for comparison. After generation, each type of handcrafted features has been exploited to feed a proper linear SVM classifier as in CSVM. Table III reports the best results obtained by tuning the related parameters of these feature extraction methods. Here again, CSVM clearly exhibits a better behavior in terms of classification accuracy and computation time.

IV. Conclusion

In this paper, we have proposed a novel CSVM network for the classification of UAV imagery. This network features the following important proprieties: 1) it exploits state-of-the art SVM classifiers as filters for feature map generation; and 2) it uses a forward supervised learning strategy for computing the weights of these filters. The results obtained on two UAV datasets acquired over urban areas related to vehicles and solar panels confirmed its efficiency compared to state-of-the-art methods in terms of accuracy and computation time. In addition, it can provide better results compared to recent solutions based on knowledge transfer from pretrained CNNs in particular when the number of training images is limited. Future directions to improve the network performances could be defined in many ways such as: 1) the exploration of other sophisticated architectures based on inception and residual

layers used in recent CNNs; 2) its extension to handle multiclass classification problems; and finally 3) the application to other classification problems with small numbers of labeled images.

ACKNOWLEDGMENT

The Authors would like to thank A. Vedaldi and K. Lenc for making available the software MatConvNet [36] used in the context of this work.

REFERENCES

[1] J. A. J. Berni, P. J. Zarco-Tejada, L. Suárez, E. Fereres, L. Suarez, and E. Fereres, "Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 3. pp. 722–738, 2009.

[2] D. Li, H. Guo, C. Wang, W. Li, H. Chen, and Z. Zuo, "Individual Tree Delineation in Windbreaks Using Airborne-Laser-Scanning Data and Unmanned Aerial Vehicle Stereo Images," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 9, pp. 1330–1334, 2016.

[3] K. Uto, H. Seki, G. Saito, and Y. Kosugi, "Characterization of rice paddies by a UAV-mounted miniature hyperspectral sensor system," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 6, no. 2, pp. 851–860, 2013.

[4] A. Y. M. Lin, A. Novo, S. Har-Noy, N. D. Ricklin, and K. Stamatiou, "Combining GeoEye-1 satellite remote sensing, UAV aerial imaging, and geophysical surveys in anomaly detection applied to archaeology," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 4, no. 4, pp. 870–876, 2011.

[5] B. Kršák *et al.*, "Use of low-cost UAV photogrammetry to analyze the accuracy of a digital elevation model in a case study," *Meas. J. Int. Meas. Confed.*, vol. 91, pp. 276–287, 2016.

[6] S. Li *et al.*, "Unsupervised Detection of Earthquake-Triggered Roof-Holes from UAV Images Using Joint Color and Shape Features," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 9, pp. 1823–1827, 2015.

[7] S. Malek, Y. Bazi, N. Alajlan, H. AlHichri, and F. Melgani, "Efficient framework for palm tree detection in UAV images," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 7, no. 12, pp. 4692–4703, Dec. 2014.

[8] K. Liu and G. Mattyus, "Fast Multiclass Vehicle Detection on Aerial Images," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 9, pp. 1938–1942, 2015.

[9] T. Moranduzzo and F. Melgani, "Detecting cars in UAV images with a catalog-based approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 10, pp. 6356–6367, 2014.

[10] T. Moranduzzo and F. Melgani, "Automatic car counting method for unmanned aerial vehicle images," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 3, pp. 1635–1647, 2014.

[11] T. Moranduzzo, F. Melgani, M. L. Mekhalfi, Y. Bazi, and N. Alajlan, "Multiclass Coarse Analysis for UAV Imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 12, pp. 6394–6406, 2015.

[12] T. Moranduzzo, F. Melgani, Y. Bazi, and N. Alajlan, "A fast object detector based on high-order gradients and Gaussian process regression for UAV images," *Int. J. Remote Sens.*, vol. 36, no. 10, pp. 2713–2733, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Arxiv.Org*, vol. 7, no. 3, pp. 171–180, 2015.

[15] C. Szegedy *et al.*, "Going Deeper with Convolutions," 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[17] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, "Object Detection Networks on Convolutional Feature Maps," vol. 8828, no. c, pp. 1–9, 2016.

[18] R. Girshick, J. Donahue, S. Member, T. Darrell, and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," vol. 38, no. 1, pp. 142–158, 2016.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks," vol. 8828, no. c, pp. 1–14, 2016.

[20] W. Ouyang *et al.*, "DeepID-Net : Object Detection with Deformable Part Based Convolutional Neural Networks," *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 39, no. 7, pp. 1320–1334, 2017.

[21] C. Couprie, L. Najman, and Y. Lecun, "for Scene Labeling," *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[22] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 39, no. 4, pp. 640–651, 2017.

[23] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[24] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Int. Conf. Learn. Represent.*, pp. 1–14, 2015.

[25] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," *Proc. Conf. BMVC*, May 2014.

[26] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of Transferability for a Generic ConvNet Representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1790–1802, 2016.

[27] R. F. Nogueira, R. de Alencar Lotufo, and R. C. Machado, "Fingerprint Liveness Detection using Convolutional Networks," *Ieee Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1206–1213, 2016.

[28] C. Gao, P. Li, Y. Zhang, J. Liu, and L. Wang, "People counting based on head detection combining Adaboost and CNN in crowded surveillance environment," *Neurocomputing*, vol. 208, pp. 1–9, 2016.

[29] D. Marmanis, M. Datcu, T. Esch, U. Stilla, and S. Member, "Using ImageNet Pretrained Networks," vol. 13, no. 1, pp. 105–109, 2016.

[30] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," *arXiv:1312.6229*, Dec. 2013.

[31] E. Othman, Y. Bazi, N. Alajlan, H. Alhichri, and F. Melgani, "Using convolutional features and a sparse autoencoder for land-use scene classification," *Int. J. Remote Sens.*, vol. 37, no. 10, pp. 1977–1995, 2016.

[32] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery," *Remote Sens.*, vol. 7, no. 11, pp. 14680–14707, Nov. 2015.

[33] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[34] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A Library for Large Linear Classification,” *J. Mach. Learn. Res.*, vol. 9, no. 2008, pp. 1871–1874, 2008.
[35] K. Chang and C. Lin, “Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines,” vol. 9, pp. 1369–1398, 2008.
[36] A. Vedaldi and C. V May, “MatConvNet Convolutional Neural Networks for MATLAB.”
[37] R. Socher, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
[38] K. He, “Deep Residual Learning (ResNet),” 2015.
[39] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” in *IEEE Conf. Comput. Vis. Pattern Recog*, 2005, pp. 886–893.
[40] S. Member and T. Ma, “Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns,” vol. 24, no. 7, pp. 971–987, 2002.
[41] S. Xu, S. Member, T. Fang, D. Li, and S. Wang, “Object Classification of Aerial Images With Bag-of-Visual Words,” vol. 7, no. 2, pp. 366–370, 2010.