SPEEDING-UP A CONVOLUTIONAL NEURAL NETWORK BY CONNECTING AN SVM NETWORK

J. Pasquet[⋆†] *M. Chaumont*[∗†] *G. Subsol* [†] *M. Derras*[⋆]

[†] LIRMM,University of Montpellier / CNRS, France
[⋆] Berger Levrault, Labège, France
[∗] University of Nîmes, France

ABSTRACT

Deep neural networks yield positive object detection results in aerial imaging. To deal with the massive computational time required, we propose to connect an *SVM Network* to the different feature maps of a CNN. After the training of this *SVM Network*, we use an *activation path* to cross the network in a predefined order. We stop the crossing as quickly as possible. This early exit from the CNN allows us to reduce the computational burden.

Experimental results are obtained for an industrial application in urban object detection. We show that potentially the computation cost could be reduced by 98%. Additionally, performance is slightly improved; for example, for a 55% recall, precision increases by 5%.

***Index Terms*—** Object Detection, Deep Convolutional Neural Network, SVM, Speed up Network

1. INTRODUCTION

In recent years, Deep Learning methods have been widely used for image classification and detection tasks [1, 2, 3]. In many challenges [4, 3], the Convolution Neural Network (CNN) method outperforms other approaches based on feature extraction and classification [5, 6], two distinctive and successive steps. However, even if Deep Learning approaches yield excellent results in detection tasks, they require significant GPU computing power and long computational times. Moreover, to analyze an image, millions of windows are tested, which requires a lot of CNN activation.

To reduce the time required to process an image, two strategies are currently available. Firstly, using classic image processing based on feature detection and classification, one can restrict the areas of the image to be analyzed by preprocessing it; however, in this method, good windows may be missed or deleted [7, 8]. Secondly, one may use any of the many cascade-based methods [9, 10] inspired by Viola and Joness seminal paper [11]. Recently, Angela et al. have proposed to increase CNN speed by using a simple cascade reject system with a deep net [12]. This method yields results similar to those obtained with classical Deep Learning approaches, and reduces computational time by a factor of 80.

In this paper, we propose an alternative approach which would enable real-time application based on an SVM cascade concept which is directly integrated into CNN layers. In this cascade approach, CNN features are given to SVMs. All these SVMs are connected in a *SVM Network* as described in [13]. The *activation path* of the CNN is then optimized by ordering the activation of the SVM nodes so as to reach a decision before crossing all the SVMs. To lower the error rate we adapt the *activation path* with a criterion of minimal number of confident SVMs.

The rest of this paper is organized as follows: first, in section 2, we present the CNN, and the integration of the *SVM Network* in the CNN. Then, in section 3, we introduce the activation path and its optimization. Further, in section 4, we present our image database and evaluate the performance of our approach. Finally, we conclude and offer future research directions.

2. INTEGRATING AN SVM NETWORK INTO A CNN

2.1. CNN architecture

The CNN architecture we use is summarized in Figure 1. We use a similar design as Krizhevsky et al.'s [3], which featured 8 layers with weights. The first 5 layers are convolutional, and the last 3 are fully connected (FC). This network contains three max pooling steps after the first, third and fifth convolutional layers. Using the ReLU function [14], non-linearity is applied to the outputs of every layer with weights. The input consists of three 8-bit images of size 64x64, which are the R, G, B channels of a color window.

In order to model the computational cost required to activate part of the network, we use the vector size of the output from a layer c, $out^{(c)}$, which can be defined as:

$$out^{(c)} = \begin{cases} N^{(c)} & for\ a\ fully\ connected\ layer \\ I_w^{(c)}.I_h^{(c)}.N^{(c)} & otherwise \end{cases} \tag{1}$$

where $I_w^{(c)}$ and $I_h^{(c)}$ are the width and the height of the input feature map for the layers $c \in \{Conv1, .., Fc2\}$, and $N^{(c)}$ is the number of neurons if c is a fully connected layer; otherwise, it represents the number of feature maps.

We then define a cost function, noted $w^{(c)}$, which sums the number of operations required to activate the c layer. In our model, we only take into account the layers with weights without any activation function, pooling or normalization cost; see Equation 2.

$$w^{(c)} = w^{(c-1)} + \begin{cases} |\mathbf{K}^{(c)}|.out^{(c-1)}.N^{(c)} & if\ convolution\ layer \\ out^{(c-1)}.N^{(c)} & if\ FC\ layer \\ 0 & otherwise \end{cases} \tag{2}$$

with $|\mathbf{K}^{(c)}|$ the kernel size of the convolution layer numbered c.

$$RC^{(c)} = \frac{w^{(c)}}{w^{(last\ layer)}} \tag{3}$$

Using equations 2 and 3, we are able to compute the relative computational cost $RC^{(c)}$ by activating the layers presented in Table 1. We note that the cost $w^{(c)}$ radically increases after the first and third convolutions. A partial crossing of the CNN should lead to a significant reduction in computational burden.

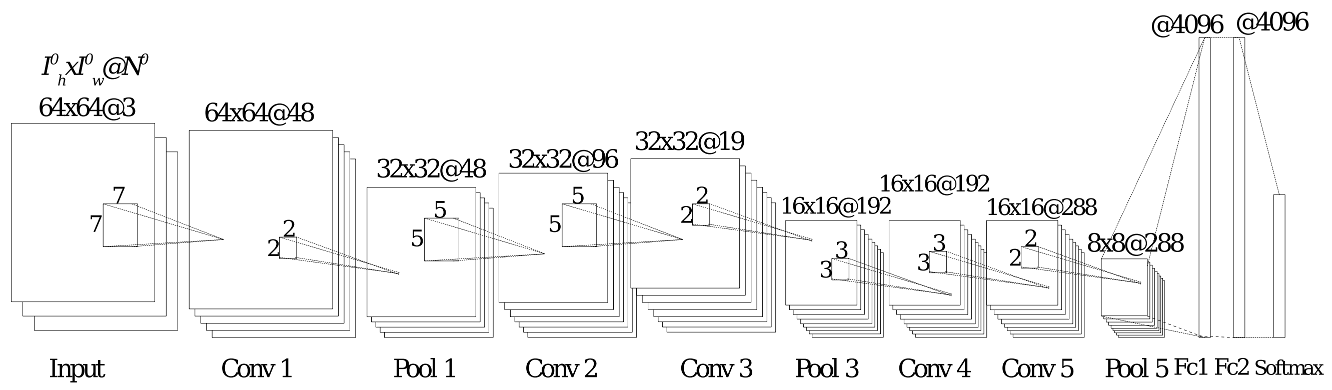

Fig. 1. Architecture of the CNN with 5 convolutions, 3 pooling and 3 fully connected layers.

layer c	$w^{(Conv1)}$	$w^{(Conv3)}$	$w^{(Conv5)}$	$w^{(Fc2)}$
$RC^{(c)}$	0.088	0.689	0.905	1

Table 1. Relative computational cost selected layers.

2.2. The SVM Network

Speeding-up the decision process is achieved by stopping the exploration of the CNN network as quickly as possible. To that effect, we integrate an *SVM Network*.

Using an SVM has been proposed to increase the efficiency of a CNN [15, 16]. In this paper, we propose to use not just one but a network of several SVMs. We define an *SVM Network* as an acyclic graph of SVM layers [13, 17]. Each SVM layer is a set of linear SVMs [18] whose inputs are the outputs of a previous layer called PL. A layer may be a CNN layer or another SVM layer. In the following, we note $S_i^{(c)}$ the input vector for the SVM numbered i, with $i \in \{0..M^{(c)}\}$, $M^{(c)}$ being the number of SVM in the layer $c \in \{Conv1, .., Fc2\} \cup \{L_{SVM1}, .., L_{SVM9}\}$.

We define three SVM layer types according to the PL layer type:

- If PL is a fully connected layer, the SVM layer will contain only one SVM. $S^{(c)}$ contains all the outputs of PL.
- If PL is an SVM layer, we randomly connect the two SVM layers. So $S^{(c)}$ is a random subset of the PL outputs.
- If PL is a convolution or pooling layer, each $S^{(c)}$ is associated to one or several PL feature maps.

The output of an SVM $o^{(c)}(S_i^{(c)})$ is defined as:

$$o^{(c)}(S_i^{(c)}) = f(\mathbf{W}_i^{(c)}.\mathbf{F}_i^{(c)}) \quad (4)$$

with $|.|$ the dot product, $\mathbf{W}_i^{(c)}$ the weights associated with the SVM i, $\mathbf{F}_i^{(c)}$ the concatenation of all feature maps in $S_i^{(c)}$ and f the sigmoid activation function.

Once an SVM layer is created, we train the next one. We repeat this step to create an *SVM Network* over the CNN.

For this work, we create nine SVM layers, as shown in Figure 2. Three SVM layers, named L_{SVM1}, L_{SVM3} and L_{SVM5} are connected with the CNN layers Pool 1, Pool 3 and Pool 5, respectively. For these SVM layers, we set $|S_i^{(c)}| \in \{1, 6\}$ $\forall i \in \{0..M^{(c)}\}$ with $c \in \{L_{SVM1}, L_{SVM3}, L_{SVM5}\}$, thus associating each feature map

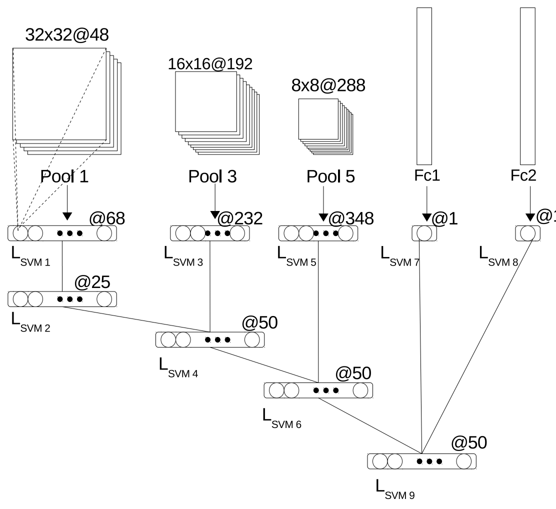

Fig. 2. Illustration of our *SVM Network* architecture. Communication with the CNN is only effective on layers 1, 3, 7, 8 and 9.

to one SVM with $|S_i^{(c)}| = 1$. For example, the layer L_{SVM1}, which is connected to Pool 1, contains 48 SVMs. Additionally, in order to improve the *SVM network*'s diversity, we also use $|S_i^{(c)}| = 6$. For example, some SVMs are connected with 6 random maps from the Pool 1. In practice, we add 20, 40, and 60 SVMs to L_{SVM1}, L_{SVM3} and L_{SVM5}, respectively. Two layers named L_{SVM7} and L_{SVM8} contain only one SVM, which uses the outputs of $FC1$ and $FC2$ as input.

The layers L_{SVM2}, L_{SVM4}, L_{SVM6}, L_{SVM9} are only connected with the outputs of SVM layers. Their input size is defined such that $|S_i^{L_{SVM2}}| = 30$, $|S_i^{L_{SVM4}}| = 50$ and $|S_i^{L_{SVM6}}| = |S_i^{L_{SVM9}}| = 70$ with $i \in \{0..M^{(c)}\}$.

Using the equation 3, we show that the relative computational cost for a SVM layer is lower than 10^{-3}. We thus ignore this small cost.

3. OPTIMIZING THE ACTIVATION PATH

The *SVM Network* is used to reduce the overall complexity of the CNN. For this purpose, we propose to optimize the *activation path* as proposed in [13]. In order to normalize the output of each SVM, we use an asymmetric sigmoid, as in equation 5.

$$o^{(c)}(S_i) = \frac{1}{1 + exp(\alpha_i^{(c)} o^{(c)}(S_i) + \beta_i^{(c)})} \quad (5)$$

with $\alpha_i^{(c)}$ and $\beta_i^{(c)}$ the scale and shift for the SVM i, in the layer c. These parameters are based on Platt's scaling algorithm [19].

During the activation of the CNN, the *SVM Network* is also activated. The following order can be used: Conv1, Pool1, L_{SVM1}, L_{SVM2}, Conv2, Conv3, Pool3, L_{SVM3}, L_{SVM4}, Conv4, Conv5, Pool5, L_{SVM5}, L_{SVM6}, FC1, L_{SVM7}, FC2, L_{SVM8}, L_{SVM9}. The goal is to stop the crossing of the acyclic CNN graph when an SVM is confident enough with its prediction.

Using a validation database, we find two thresholds for each SVM. For a binary classification, we note $\theta^{(A)}$ the threshold needed to classify an object as $classA$, and $\theta^{(B)}$, respectively, for $classB$ classification. These two parameters are obtained by maximizing the recall such that the minimal precision is higher than p_{min}, as in equation 6.

$$\theta_i^{(s)} = \underset{\theta_i^{(s)} \in [0..1]}{\arg\max}\, Recall_i^{(s)}(\theta) \text{ such that } Precison_i^{(s)}(\theta) \geq p_{min}^{(s)} \quad (6)$$

with $s \in \{classA, classB\}$, and $Recall$ and $Precision$ the functions giving the recall and precision of SVM i for class s.

On each layer, we use the recall to sort the SVMs in descending order to maximize the number of rejected or accepted window. As a result, the SVM with the best recall and achieving minimal precision is used first, and so forth. This order defines the *activation path*. SVMs with no recall with the minimal precision required are removed from the *activation path*.

During the evaluation step, several million windows are tested. Thus, a large number of false positives is generated for the *SVM Network*. Indeed, the global error is the sum of errors for each SVM in the path. In order to reduce the false positive rate, and to give more robustness to the rejection process, we modify the reject condition that allows us to stop the crossing. During the test, and therefore during the crossing of the CNN and SVM network, instead of relying on just one SVM decision, we pool the SVMs decisions from all the nodes already crossed such that a sufficient number of SVMs are confident for classifying objects into class A or class B.

We therefore add a parameter to our *activation path* which we call the pooling decisions, noted as P. When an SVM categorizes the window as a class A or B object, the decision P is shifted towards this class. Equation 7 shows the evolution of the P parameter during the crossing of the node i.

$$\begin{cases} o_i > \theta_i^{(A)} & \rightarrow P = P + 1 \\ o_i < \theta_i^{(B)} & \rightarrow P = P - 1 \end{cases}$$

$$\begin{cases} P > +O_{lim} & \rightarrow \text{stop the exploration returns the class A} \\ P < -O_{lim} & \rightarrow \text{stop the exploration returns the class B} \end{cases} \quad (7)$$

Where O_{lim} is the criterion of minimal number of confident SVMs.

We stop crossing the network when the pooling decision P is greater than O_{lim}. To obtain an optimal performance, we need to set a large enough O_{lim} criterion. However, if this criterion increased, we would assume the stop condition would become increasingly rare. Accordingly, as the number of activated SVMs increases, so does complexity.

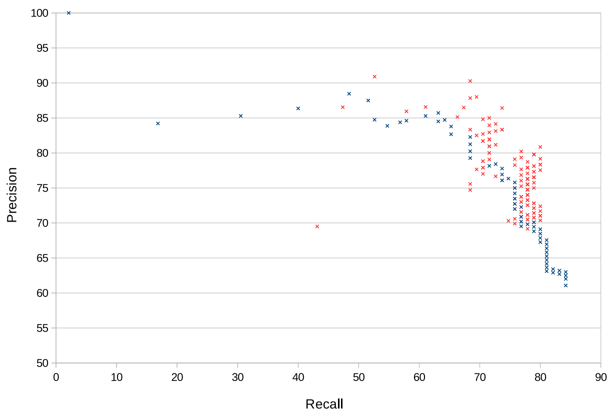

Fig. 3. The red points represent the ROC curve for the SVM_{FC2}, and the blue points trace the efficiency of the *activation path* according to the O_{lim} criterion.

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

We focused on detecting tombs in cemeteries with high-definition aerial images for geo-localization and digital heritage purposes [20]. Tomb detection is a challenging problem, as tombs substantially vary in appearance, color, size and disposition on aerial images. Moreover, vegetation, as well as the shadows cast by buildings, pedestrians or utility vehicles create multiple distortions and occlusions in the images. Lastly, this type of detection is known as multiple-object detection, as cemeteries typically contain hundreds or thousands of tombs.

For the training database, we used 19 aerial images of cemeteries of French villages provided by the Berger Levrault company[1], and which showed about 4,500 tombs. For the validation database, we used 3 images containing about 750 tombs.

The algorithm returned a list of bounding boxes with SVM classification scores. For the CNN approach, after the training step, we replaced the softmax layer by an SVM [21, 22]. This SVM, noted SVM_{FC2}, predicted from the FC2 features. For our SVN network method, the scores were 1 for the tombs and 0 for other objects. The *activation path* was built using $p_{min}^{tombs} > 0.99$ and $p_{min}^{others} > 0.995$.

Evaluation was performed according to this list. The ground truth contained 700 tombs from the images of 2 other cemeteries. According to the PASCAL VOC evaluation protocol [4], a detected bounding box matches if the overlap with the ground truth is larger than 50%.

In the learning and testing step, we used the Caffe framework for CNN [23]. After the CNN training, we extract the features from different layers in order to train the SVMs. Then, for each of them we add a fully connected layer to our Caffe model using the SVM weights. At the moment, the whole CNN is still activated. Nevertheless, we check the outputs of the SVM layers to see if potentially the network would have taken its decision before. We will discuss implementation issues to get a real acceleration in the last section.

4.2. Results Analysis

As shown in Figure 3, the *SVM Network*'s precision is greater than that of the SVM_{FC2} for recall ranging from 50 to 70. For a recall

[1]Berger-Levrault is a French public regulation expert that addresses healthcare and local public administrations, www.berger-levrault.com

ranging between 70 and 80, both achieve a high precision of approximately 75%.

However, in the single instance where O_{lim} criterion is set to 0, we observe that the SVM_{FC2} outperforms the *SVM Network* with 87% precision when the chosen recall is 43.1. In this case, the *activation path* thus rejects or accepts a window without any confirmation. According to the predefined p_{min}, each SVM has a minimal precision of 99.5%. During the testing step, when millions of windows are assessed, this error cannot be ignored. Table 2 shows that the precision increases with the O_{lim} criterion from 0 to 2. However, beyond this value, precision remains flat because some tombs are detected by only a few SVMs in the *activation path*. Therefore, the number of SVMs which are able to detect these tombs is lower than the O_{lim} criterion.

O_{lim}	Recall	Precision	% RC
0	43.15	69.4915	1.38
1	47.36	86.5385	1.59
2	52.63	90.90	1.75
10	70.52	84.81	2.72
11	70.52	82.7	2.83
26	71.57	80.95	4.74
27	70.52	77.90	4.97

Table 2. Performance of the *SVM Network* according to O_{lim} criterion in the test database.

Table 2 also gives the average computational cost needed to activate the CNN according to the equation 2. As the data shows, only 1.75% of the network is used for a minimal number of confident SVMs lower than 3. Moreover, the maximum cost is 2.83% for a criterion lower than 12. The *activation path* could potentially allow us to save up to 98.25% of processing time.

An SVM is activated when it rejects or accepts the window. Figure 5 shows the activity for each SVM inside the *SVM Network*. We observe that the entire *SVM Network* is used to predict all sliding windows of an image. Furthermore, we note from table 3 that SVMs from layers L_{SVM1} and L_{SVM2} have less than 0.4% of false positive. Windows with hard to predict image content activate more and more SVMs, so that the number of false positives is increased after the first layers. Table 3 summarizes the histogram in Figure 5. The error for the two-first stage SVM layers is very low (about 0.3%), but they only detect 0.02% of the tombs. These layers significantly filter other objects. The more complex an object is, the deeper SVMs will classify it, and, in turn, the deeper the SVM, the greater the number of false positives. Finally, on the last SVM layers the classification rate is improved, with 35% of true tombs for 30% of FP; however, the CNN network is fully activated and there is no complexity gain. In this case, the network is similar in performance and computational cost to an SVM_{FC2}.

SVM Layers	#Activations	Error (%)	FP (%)	TP (%)
$L_{SVM\{1,2\}}$	3,946,414	0.38	0.38	0.02
$L_{SVM\{3,4\}}$	31,433	18.76	18.61	0.71
$L_{SVM\{5,6\}}$	3,724	50.70	49.03	5.89
$L_{SVM\{7,8\,and\,9\}}$	450	42.85	30.84	35.61

Table 3. Average efficiency for groups of SVM layers with the criterion of minimal number of confident SVMs set to 2.

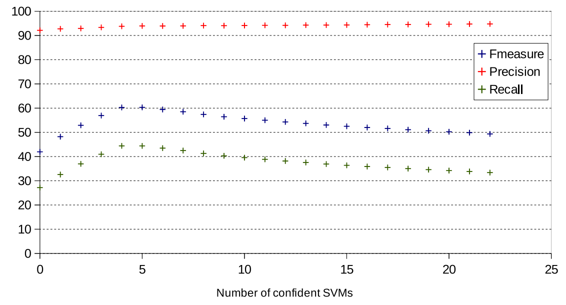

Fig. 4. *Activation path* performance according to O_{lim} criterion.

4.3. Optimal Activation Path

To find the most optimal O_{lim}, we use a validation database. In this database, we try to optimize one parameter out of several: precision, recall, F-measure or computational cost. Figure 4 gives $O_{lim} = 5$, maximizing the F-measure to 60%. Following evaluation, we obtain a precision of 90% and a recall around 68.4%. For the same recall the SVM_{FC2} reaches 80% in precision. Moreover, in this condition, the SVM Network only uses 2.17% of the CNN.

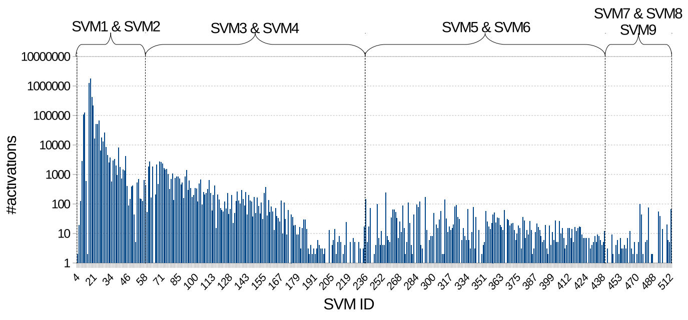

Fig. 5. On this histogram, each vertical blue line represents the number of activations for each SVM in the *activation path* during the testing step.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented a new solution to add an adaptive *SVM Network* to a CNN. The tuning of the *activation path* allowed us to obtain a 9% precision gain for a recall set to 67%. Moreover, the *activation path* massively reduced the computational cost required. In our experiments, for a recall of only 67%, an average of 97.8% of the network remained unused.

At the moment, the CNN is deployed on a GPU, and requires to be fed with a batch of images in order to benefit of massive parallelism. The difficulty is that all images from a batch are rejected or accepted by different SVMs in the activation path. When an image from the batch is rejected by an SVM from the activation path the batch size decreases for the next layer. We are thus unable to ensure a constant size of the batches during the network crossing. The current implementation does not use totally the parallelism possibilities. This could be obtained using a trick for keeping batch sizes constant or using batch sizes of predefined different sizes for each layer.

Connecting the SVM network to the entire CNN - and not just to the pooling layer outputs - could offer interesting perspectives for future work. Moreover, weighing the pooling decision during activation could help reduce the number of false positives on the first activated SVMs. Finally, another possible way to increase overall performance would be to extend the activation path to a multiclass problem.

6. REFERENCES

[1] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[2] J. Leitloff, S. Hinz, and U. Stilla, "Vehicle detection in very high resolution satellite images of city areas," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 48, no. 7, pp. 2795–2806, July 2010.

[3] I. Sutskever A. Krizhevsky 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.

[4] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, June 2005, vol. 1, pp. 886–893.

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

[7] J. R. R. Uijlings, K. E. A. Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[8] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[9] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 2903–2910.

[10] Qiang Zhu, Qiang Zhu, Shai Avidan, Shai Avidan, Mei chen Yeh, Mei chen Yeh, Kwang ting Cheng, and Kwang ting Cheng, "Fast human detection using a cascade of Histograms of Oriented Gradients," in *CVPR06*, 2006, pp. 1491–1498.

[11] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, vol. 1, pp. I–511–I–518.

[12] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson, "Real-time pedestrian detection with deep network cascades," in *Proceedings of BMVC 2015*, 2015.

[13] Pasquet J., Chaumont M., Subsol G., and Derras M., "An efficient multi-resolution SVM network approach for object detection in aerial images," in Machine Learning for signal processing, MLSP2015. IEEE, 2015.

[14] Vinod Nair and Geoffrey E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Johannes Frnkranz and Thorsten Joachims, Eds. 2010, pp. 807–814, Omnipress.

[15] Yichuan Tang, "Deep learning using support vector machines," *Computing Research Repository (CoRR)*, vol. abs/1306.0239, 2013.

[16] Haigang Zhu, Xiaogang Chen, Weiqun Dai, Kun Fu, Qixiang Ye, and Jianbin Jiao, "Orientation robust object detection in aerial images using deep convolutional neural network," in *Image Processing (ICIP), 2015 IEEE International Conference on*, Sept 2015, pp. 3735–3739.

[17] Pasquet J., Chaumont M., Subsol G., and Derras M., "Optimizing color information processing inside an svm network," in *Visual Information Processing and Communication Conferences, Electronic Imaging*. IEEE, Feb 2016.

[18] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[19] Jing Gao and Pang ning Tan, "Converting output scores from outlier detection algorithms into probability estimates," in *In Proc. of the Sixth IEEE Int. Conf. on Data Mining (ICDM06)*, 2006, pp. 212–221.

[20] M. Chaumont, L. Tribouillard, G. Subsol, F. Courtade, J. Pasquet, and M. Derras, "Automatic localization of tombs in aerial imagery: Application to the digital archiving of cemetery heritage," in *Digital Heritage International Congress (DigitalHeritage), 2013*, Oct 2013, vol. 1, pp. 657–660.

[21] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International Conference on Machine Learning (ICML)*, June 2014.

[22] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[23] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.