Graph Embedded Convolutional Neural Networks in Human Crowd Detection for Drone Flight Safety

Maria Tzelepi and Anastasios Tefas
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
Email: mtzelepi@csd.auth.gr, tefas@aiia.csd.auth.gr

Abstract—In this paper, we propose a novel human crowd detection method that uses deep Convolutional Neural Networks for drone flight safety purposes. The first contribution of this work is to provide lightweight architectures, as restricted by the computational capacity of the specific application, capable of effectively distinguishing between crowded and non-crowded scenes from drone-captured images, and provide crowd heatmaps which can be used to semantically enrich the flight maps by defining no-fly zones. The second contribution of this work is to propose a novel generic regularization technique, based on the Graph Embedding framework, applicable to different deep architectures for generic classification problems. The experimental validation is performed on a new dataset constructed for the task of human crowd detection from drone-captured images, and indicates the effectiveness of the proposed detector, as well as of the proposed regularizers in terms of classification accuracy. Finally, since the proposed regularization scheme is applicable in generic classification problems, we have also conducted experiments on two additional datasets, where the enhanced performance of the regularizers is also validated.

Index Terms—Drones, Crowd detection, Deep Learning, Regularization, Graph Embedding, Convolutional Neural Networks.

1 Introduction

The recent introduction of drones in a wide spectrum of applications including rescue, surveillance, and entertainment, is accompanied by the requirement of their safe operation. Except for the robustness to failures, a primary step to settle the issue of safety constitutes in defining no-fly zones for crowd avoidance, since a drone may fly in vicinity of crowds, and is potentially exposed to environmental hazards or unpredictable errors and malfunctions, that render the emergency landing inevitable. Furthermore, drone flight regulations in several Countries' national legislation, especially in European Countries, request a safe distance to be maintained between the drone and crowds, and the drone to never fly over a crowd. For example, the drone flight regulation rules for UK[1] define that drones should not be flown within 50m of people and within 150m of a crowd of over 1000 people, in Italy[2] it is not allowed for the drone to operate at a distance less than 50m, whereas in Germany[3] it is prohibitive for a drone to operate at a distance of less than 100m from crowds (where an assemblage of more than 12 individuals is defined as crowd). Therefore, it is crucial for the drone to be capable of detecting crowds in order to define no-fly zones and proceed to re-planning during the flying operation. This could also allow for diminishing the restrictions for autonomous flying of drones ensuring the crowds' safety in the flying area. Towards this end, in this work we deal with the problem of crowd detection from drones, for crowd avoidance using the state-of-the-art deep Convolutional Neural Networks (CNN), [1], [2].

During the recent years, deep CNNs have been proven as one of the most promising research directions in computer vision, providing significant results in a plethora of computer vision tasks. More specifically, deep CNNs have been successfully applied in image classification [3], [4], object detection [5], [6], [7], face recognition [8], image retrieval [9], digit recognition [10], [11], pose estimation [12], pedestrian detection [13], and scene recognition [14]. The major reasons underlying their success are the availability of large annotated datasets, and the Graphics Processing Units (GPUs) computational power and affordability.

Deep CNNs consist of a number of convolutional and sub-sampling layers with non-linear activations, usually followed by fully connected layers. That is, the neural network accepts as input a three dimensional tensor with dimensions (i.e., width and height) equal to the dimensions of the input image, and depth equal to the number of color channels (that is three in RGB images). Three dimensional filters/kernels are learned and applied in each layer where convolution is performed and the output is propagated to the neurons of the subsequent layer for non-linear transformation, using appropriate activation functions. After a series of convolutional and sub-sampling layers, the structure changes to fully connected layers and single dimensional signals. These activations are usually used as feature representations in classification, retrieval, and clustering.

In this paper, we propose a crowd detection method for drone flight safety purposes that utilizes fully convolutional deep CNNs. That is, we predict for each patch a probability of crowd existence, and then we provide a semantic heatmap of

1. http://www.caa.co.uk/Commercial-industry/Aircraft/Unmanned-aircraft/Small-drones/Permissions-and-exemptions-for-commercial-work-involving-small-drones/

2. http://www.enac.gov.it/repository/ContentManagement/information/N1220929004/Regulation_RPAS_Issue_2_Rev%202_eng.pdf

3. http://www.lba.de

the estimated probability of existence of crowd in each location within the captured scene, instead of a box surrounding the crowded area. Therefore, we describe the whole procedure as detection. We aim to provide a lightweight deep CNN model, as restricted by the computational capacity of the specific application, that can effectively distinguish between crowded and non-crowded scenes, in drone-captured images, and provide semantic heatmaps that can be used to semantically augment the flying zones. The fully convolutional nature of the proposed model is crucial in handling input images with arbitrary dimension, and estimating a heatmap for the crowded areas. This will allow for semantically annotating the corresponding maps and defining no-fly zones. Furthermore, this will allow for handling low computational and memory resources on-drone whenever other processes occur (*e.g.*, re-planning, SLAM, etc.), and only low-dimensional images can be processed on the fly for crowd avoidance.

Additionally, a principal direction of this work is to propose regularization techniques in order to prevent overfitting and enhance the generalization ability of our model. Generally, this constitutes a major issue in deep learning algorithms, since neural networks are prone to overfitting due to their high capacity. To this end, we propose a regularization scheme based on the Graph Embedding (GE) framework [15]. Recently, the Extreme Learning Machine algorithm [16], as well as a novel dimensionality reduction algorithm that uses the Mutual Information as an optimization criterion [17], have been successfully integrated in the GE framework. Thus, in this work we aim to introduce the ideas described in the GE framework in Deep Learning, by proposing a multiple-loss architecture as a regularized training method. The above method is generic and can be applied in several deep learning architectures for classification purposes. Note also, that the proposed regularization scheme acts in activation level, as compared with other regularizers, like the standard L1/L2 ones, which act in weights level, penalizing large weights during the network optimization. The experimental evaluation on the constructed drone crowd dataset, as well as on two additional classification datasets, validates the effectiveness of the proposed regularizers in accomplishing better performance in terms of accuracy.

Generally, multitask-learning [18] constitutes a way of improving the generalization performance of a model. In [19] the authors introduced techniques emerged in semi-supervised learning into the deep learning domain. That is, they combined a supervised learner with an unsupervised regularizer performing semi-supervised learning. Subsequently, in [20] stimulated training of deep neural networks for network regularization and robust adaptation is investigated. More specifically, first stimulated deep neural network regularization is applied to large vocabulary recognition tasks, achieving consistent gains, and second, based on the activation patterns obtained by stimulated learning, a smoothing approach is proposed to regularize the deep neural network adaptation schemes. Finally, in [21] contemporaneous with our work, the authors propose a framework for activation regularization aiming to improve interpretability and regularization in deep neural models. Specifically, the framework allows appropriate target patterns to be interpreted on activation function outputs. The target patterns are introduced to induce desired information such as smoothness, and a regularization term is added to the cost function, encouraging activation outputs to perform similarly to the target pattern.

The main contributions of this paper can be summarized as follows:

- A fully convolutional deep neural model for crowd detection from drones towards crowd avoidance is proposed.
- A novel regularization scheme, based on the GE framework, applicable to different deep architectures for generic classification problems is proposed.
- A new drone crowd dataset has been constructed, in order to validate the proposed method since there is no such dataset publicly available.

The remainder of the manuscript is structured as follows. Section 2 discusses the related work. The proposed fully convolutional model for crowd detection, a brief review of GE framework, as well as the proposed graph embedded CNN are presented in detail in Section 3. In Section 4 we present the constructed drone crowd dataset. Subsequently, in Section 5 we provide the implementation details of the proposed method as well as the experimental evaluation of our method. Finally, the conclusions are drawn in Section 6.

2 Related Work

To the best of our knowledge, crowd detection in images captured from drones is an uncharted territory. A first attempt utilizing the state-of-art deep CNNs is presented in [22], where a CNN pretrained model is finetuned for the task of crowd detection, proposing a two-loss architecture. In this section we review the existing literature regarding the crowd detection task from images that are not captured by drones, while we also briefly refer to research works in the general field of crowd analysis that utilize deep learning techniques.

Even though there are a lot of works in the crowd analysis domain (*e.g.* crowd counting [23], abnormal crowd behaviour analysis [24] etc.), the research in crowd detection is very limited, since these works consider crowded scenes. In [25], the authors propose a crowd detection method based on spatiotemporal analysis of the video sequence. Subsequently, an algorithm for crowd detection in still images, utilizing a statistical, Poisson model of occurrences of quantized SIFT words across an image [26], is proposed. Finally, in [27] the authors proposed texture classification methods for detecting high density crowds in aerial non-drone images.

Over the last few years, several works that use deep CNNs have been emerged in the field of crowd counting, analysis and understanding. A deep learning framework for crowd density estimation in extemely dense crowd images, is proposed in [28]. In [29] the authors also propose a model, trained with two related learning objectives for crowd counting, while in [30] the authors propose a Multi-column Convolutional Neural Network architecture to estimate the crowd count. Subsequently, in [31] the authors develop a multitask deep model to jointly learn and combine appearance and motion features towards crowd understanding, while in [32] a crowd abnormal event detection method using deep CNNs is proposed. In [33] a switching CNN that leverages variations of crowd density in an image in order to enhance the accuracy and localization of the estimated crowd count, is proposed. In [34] a method for computing an estimation of the amount of individuals in highly dense crowd images relying on multiple sources, like low confidence head detections, repetition of texture elements, and frequency-domain analysis to estimate counts, is proposed.

Subsequently, in [35], the authors address the problem of human detection in dense crowd, exploring the utilization of spatial or contextual constraints for improving human detection. In [36] a tracker capable of tracking hundreds of people in extremely dense crowds is proposed, by formulating online crowd tracking as Binary Quadratic Programing. Subsequently, in [37] a deep learning approach to detect real-world anomalies in surveillance videos is proposed, exploiting both normal and anomalous surveillance videos. Finally, in [38] a new approach for autonomous navigation for low-altitude UAVs in urban areas is proposed. The method for a given mission computes safe waypoints, that dynamically adapt the flight plan to the UAVs surroundings by avoiding objects like cars or pedestrians. However, we note that all these works do not consider images captured from drones, and they also consider crowded scenes. Thus, since the crowd first need to be detected, this highlights the demand for algorithms able to efficiently distinguish between crowded and non-crowded scenes.

3 Proposed Method

In this work, we propose a deep learning regularization technique utilizing the concepts described in the GE framework, for human crowd detection in images captured from drones. To this aim, we utilize the deep CNNs. Specifically, we train a fully convolutional neural model.

3.1 Fully Convolutional Neural Network

The underlying reasons behind the fully convolutional architecture are presented below. Firstly, the convolutional neural layers preserve spatial information due to the spatial arrangement of the activations, in opposition to the fully connected layers that discard it since they are connected to all the input neurons. That is, the convolutional layers inherently produce feature maps with spatial information. This attribute is generally beneficial in a series of computer vision tasks. For example, in [9], [39] it is shown that the utilization of the convolutional layers for feature extraction provides superior performance in the retrieval task. In this way, the proposed classifier also benefits from the convolutional layers, since it decides on the crowd existence based on the convolutional features. Additionally, an architecture without fully connected layers drastically decreases the amount of the model parameters, and therefore the computational cost is restricted, since the fully connected layers of deep CNNs usually occupy the most of the model parameters. For example, in VGG [40] the fully connected layers comprise 102M parameters out of a total of 138M parameters, while in AlexNet [3] they occupy 59M out of the 61M parameters. Furthermore, this also allows arbitrary-sized input images, as the fixed-length input demand concerns the fully connected layers. As a result, this allows for using low-resolution images, that can be proven beneficial in the specific application, since it can further restrict the computational cost. We should note that the fully convolutional choice does not affect the memory bandwidth needed, however it allows for adjusting the complexity, e.g. with stride, which is not feasible in models with fully connected layers. Finally, we note that state-of-the-art object detectors, like SSD [6], and YOLO9000 [7] also use fully convolutional architectures.

3.2 Graph Embedding and Dimensionality Reduction

GE [15] unifies a series of dimensionality reduction algorithms within a common optimization scheme. That is, in GE each algorithm can be considered as the direct graph embedding that describes certain desired statistical or geometrical properties of the dataset. More specifically, let $\boldsymbol{\mathcal{G}} = \{\boldsymbol{X}, \boldsymbol{W}\}$ be an undirected weighted graph, with vertex set the data points $\boldsymbol{x}_i \in \mathbb{R}^d$ arranged in a data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, , \ldots, \boldsymbol{x}_N]$, where N is the number of samples, d is the feature dimension, and $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ is the similarity matrix. The GE of the graph $\boldsymbol{\mathcal{G}}$ is then defined as an algorithm to find the desired low dimensional representation of the data that best preserves the relationships between the vertex pairs of $\boldsymbol{\mathcal{G}}$. The graph $\boldsymbol{\mathcal{G}}$ can be seen as an intrinsic graph. Additionally, a penalty graph $\boldsymbol{G}^p = \{\boldsymbol{X}, \boldsymbol{W}^p\}$ can also be defined, such that the weight matrix, $\boldsymbol{W}^p \in \mathbb{R}^{N \times N}$, of the graph penalizes specific characteristics of the data structure. For simplicity, we present the one-dimensional case, assuming that $\boldsymbol{y} = [y_1, , \ldots, y_N]^T$ is the vector containing at each element y_i the projection of each data sample $\boldsymbol{x}_i$.
Thus, the generic graph criterion to be optimized is:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}^T \boldsymbol{B} \boldsymbol{y} = q}{\arg\min} \sum_{\substack{i,j=1, \\ i \neq j}}^{N} \|y_i - y_j\|_2^2 \, W_{ij} = \underset{\boldsymbol{y}^T \boldsymbol{B} \boldsymbol{y} = q}{\arg\min} \boldsymbol{y}^T \boldsymbol{L} \boldsymbol{y}, \tag{1}$$

where $\boldsymbol{L}$ is the graph Laplacian defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, $\boldsymbol{D}$ is the diagonal degree matrix defined as $D_{ii} = \sum_{i \neq j} W_{ij}, \forall i$, $\boldsymbol{B}$ is the constraint matrix to avoid trivial solutions and is typically a diagonal matrix for scale normalization, or the graph Laplacian of $\boldsymbol{G}^p$, that is $\boldsymbol{B} = \boldsymbol{L}^p = \boldsymbol{D}^p - \boldsymbol{W}^p$, and q is a constant. The matrices $\boldsymbol{W}$ and $\boldsymbol{B}$ allow to formulate different dimensionality reduction algorithms.

If we assume that the vector $\boldsymbol{y}$ is obtained by the linear projection $\boldsymbol{y} = \boldsymbol{X}^T \boldsymbol{w}$, where $\boldsymbol{w} \in \mathbb{R}^d$ is the projection vector, then the objective becomes:

$$\boldsymbol{w}^* = \underset{\substack{\boldsymbol{w}^T \boldsymbol{X} \boldsymbol{B} \boldsymbol{X}^T \boldsymbol{w} = q, \\ or \ \boldsymbol{w}^T \boldsymbol{w} = q}}{\arg\min} \sum_{\substack{i,j=1, \\ i \neq j}}^{N} \|\boldsymbol{w}^T \boldsymbol{x}_i - \boldsymbol{w}^T \boldsymbol{x}_j\|_2^2 \, W_{ij} = \underset{\substack{\boldsymbol{w}^T \boldsymbol{X} \boldsymbol{B} \boldsymbol{X}^T \boldsymbol{w} = q, \\ or \ \boldsymbol{w}^T \boldsymbol{w} = q}}{\arg\min} \boldsymbol{w}^T \boldsymbol{X} \boldsymbol{L} \boldsymbol{X}^T \boldsymbol{w} \tag{2}$$

The Principal Component Analysis (PCA) [41] seeks for a projection that maximizes the variance of the data. That is, it finds and removes the projection direction with minimal variance, i.e.,

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}^T \boldsymbol{w} = 1}{\arg\min} \boldsymbol{w}^T \boldsymbol{C} \boldsymbol{w}, \tag{3}$$

with

$$\boldsymbol{C} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T = \frac{1}{N} \boldsymbol{X} (\boldsymbol{I} - \frac{1}{N} \boldsymbol{e} \boldsymbol{e}^T) \boldsymbol{X}^T, \tag{4}$$

where $\boldsymbol{e}$ is an N-dimensional vector with ones, $\boldsymbol{I}$ is an identity matrix, $\boldsymbol{C}$ is the covariance matrix and $\bar{\boldsymbol{x}}$ is the mean of all samples. In graph embedding, the intrinsic graph characterizes the properties of the projections that need to be found and discarded, that is, the directions of small variance in PCA. Thus, over the GE view, the similarity matrix $\boldsymbol{W}$ and and the constraint matrix $\boldsymbol{B}$ are as follows for the PCA algorithm: $W_{ij} = \frac{1}{N}, i \neq j, \boldsymbol{B} = \boldsymbol{I}$.

The Linear Discriminant Analysis (LDA) [42] algorithm seeks for the most discriminative projection directions by minimizing the ratio between the intraclass and interclass scatters:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}^T\boldsymbol{S}_B\boldsymbol{W}=d} \boldsymbol{w}^T\boldsymbol{S}_W\boldsymbol{w} = \arg\min_{w} \frac{\boldsymbol{w}^T\boldsymbol{S}_W\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{S}_B\boldsymbol{w}} = \arg\min_{w} \frac{\boldsymbol{w}^T\boldsymbol{S}_W\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{C}\boldsymbol{w}}, \quad (5)$$

with

$$\boldsymbol{S_W} = \sum_{i=1}^{N}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^{c_i})(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^{c_i})^T = \boldsymbol{X}(\boldsymbol{I} - \sum_{c=1}^{N_c}\frac{1}{n_c}\boldsymbol{e}^c\boldsymbol{e}^{cT})\boldsymbol{X}^T, \quad (6)$$

$$\boldsymbol{S_B} = \sum_{i=1}^{N} n_c(\bar{\boldsymbol{x}}^c - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}^c - \bar{\boldsymbol{x}})^T = N\boldsymbol{C} - \boldsymbol{S}\boldsymbol{w}, \quad (7)$$

where $\bar{\boldsymbol{x}}^c$ is the mean of the c-th class, and $\boldsymbol{e}^c$ is an N-dimensional vector with $\boldsymbol{e}^c(i) = 1$, if $c = c_i$ and 0 otherwise.

Hence, the similarity matrix $\boldsymbol{W}$ and and the constraint matrix $\boldsymbol{B}$ are as follows for the LDA algorithm: $W_{ij} = \frac{\delta_{l_i,l_j}}{n_{l_i}}, \boldsymbol{B} = \boldsymbol{I} - \frac{1}{N}\boldsymbol{e}\boldsymbol{e}^T$.

3.3 Graph Embedded CNN

In this work, considering a deep neural architecture for classification purposes with a *Softmax Loss*, we propose to attach one or multiple additional Loss layers which impose certain constraints, motivated by the GE algorithms. The *Euclidean Loss* (sum of squares) layer can be easily employed to implement them, however more sophisticated losses can also be defined. We should also note that in this work we utilize the widely used Softmax Loss layer in deep neural networks [40], [43], [3], however the Hinge Loss layer could also utilized for the classification task.

Thus, for a set of N input images $\mathcal{X} = \{\mathbf{X}_i, i = 1, \ldots, N\}$ we consider the corresponding representations, $\mathbf{y}_i^L$, of the feature space generated by a specific deep neural layer, L, as the corresponding projection of training data in the GE concept. Then, the Softmax Loss is defined as:

$$L_s = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} l_{i,k}log(p_{i,k}), \quad (8)$$

where K is the number of classes, $l_{i,k} \in \{0, 1\}$ is a binary indicator that takes the value 1 if the class label k is the correct classification for the sample i, and $p_{i,k}$ is the predicted softmax probability the sample i to belong to the class k.

The Euclidean loss for a target representation $\boldsymbol{t}_i$ which is determined by a specific GE algorithm is defined as:

$$L_e = \frac{1}{2N}\sum_{i=1}^{N}\|\boldsymbol{y}_i^L - \boldsymbol{t}_i\|_2^2 \quad (9)$$

We note that the Euclidean loss can be attached to all the deep neural layers. Thus, for a deep model of N_L layers, the total loss in the regularized training scheme is computed by summing the above losses:

$$L = L_s + \sum_{i=1}^{N_L}\eta_i L_e, \quad (10)$$

where the parameter $\eta_i \in [0, 1]$ controls the relative importance of the Euclidean loss of each deep layer, and $\eta_i = 0$ means that no regularizer is attached to the i-th layer. The Euclidean loss layer acts as a regularizer, which can transfuse the ideas of GE in deep learning. It can be attached to all the layers of the deep architecture, to individual layers, as well as to combinations of layers. Furthermore, different regularizers, motivated by different GE approaches, can be simultaneously applied to a deep neural layer.

In the following subsection we describe the Discriminant Analysis (DA) regularization approach and the Minimum Enclosed Ball (MEB) regularization, and we accordingly define their objectives using appropriate target representations in the Euclidean Loss layer. We note that the proposed regularization method is straightforward, and can materialize all the GE algorithms (e.g. the Marginal Fisher Analysis [15], etc.).

3.3.1 Discriminant Analysis Regularization

Inspired by the LDA method, that aims at best separating training samples of different classes, by projecting them into a new low-dimensional space, that maximizes the between-class separability while minimizing their within-class variability, we propose a new regularized training method. Thus, the new model, except for the softmax loss layer that preserves the between class separability, includes a Euclidean loss layer that aims to bring the training samples of the same class closer to the class centroid.

That is, considering a labeled representation $(\mathbf{y}_i^L, l_i)$, where $\mathbf{y}_i^L$ is the image representation and l_i is the corresponding image label, we aim to minimize the squared distance between $\mathbf{y}_i^L$ and the mean representation of its class.

Let $\mathcal{X} = \{\mathbf{X}_i, i = 1, \ldots, N\}$ be the set of N images of the training set, $\mathcal{Y}^L = \{\mathbf{y}_i^L, i = 1, \ldots, N\}$ the set of N feature representations emerged in the L layer of a deep neural model, and $C^i = \{\mathbf{c}_k, k = 1, \ldots, K^i\}$ the set of K^i representations of the i-th image, belonging to the same class. We compute the mean vector of the K^i representations of C^i to the certain image representation $\mathbf{y}_i^L$, and we denote it by $\boldsymbol{\mu}_c^i$. That is, $\boldsymbol{\mu}_c^i = \frac{1}{K^i}\sum_k \mathbf{c}_k$.

Then our goal is defined by the following optimization problem:

$$\min_{\mathbf{y}_i^L \in \mathcal{Y}^L} \mathcal{J}_{DA} = \min_{\mathbf{y}_i^L \in \mathcal{Y}^L} \sum_{i=1}^{N}\|\mathbf{y}_i^L - \boldsymbol{\mu}_c^i\|_2^2, \quad (11)$$

Thus, the total loss is formulated as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} l_{i,k}log(p_{i,k}) + \frac{1}{2N}\sum_{i=1}^{N}\|\mathbf{y}_i^L - \boldsymbol{\mu}_c^i\|_2^2 \quad (12)$$

We solve the above optimization problem using gradient descent. It is noted that the proposed regularizer can be implemented over the entire dataset, for the mean vectors of all the samples belonging to one class, as well as in terms of mini-batch training. In our experiments we implement it in terms of mini-batch training. That is, the mean vectors of each class are computed for each batch of N_b training samples.

3.3.2 Minimum Enclosing Ball Regularization

In this regularization approach, we apply an objective which aims at finding a projection that minimizes the variance among the training samples, and we abbreviate it as MEB. The rationale behind this idea is rooted in the radius-margin based Support Vector Machines (SVM) [44], [45], [46]. More specifically, in [47], it is stated that the generalization error bound of the max-margin SVMs depends on not only the squared separating margin, γ^2, of the positive/negative training samples, but on

the radius-margin ratio, R^2/γ^2, where R is the radius of the minimum enclosing ball (MEB) of all the training samples. For a fixed feature space, the dependency of the error bound on the radius can be ignored in the optimization procedure, since the radius, R, is constant. However, when R is determined by the MEB of the training data, the model has the risk that the margin can be increased by simply expanding the MEB of the training data in the feature space. In order to remedy this problem, an algorithm that optimizes the error bound taking account of both the margin and the radius, in the context of Multiple Kernel Learning, is proposed in [45]. In [48], the authors also propose to incorporate a radius-margin bound as a regularization term into a deep model for $3D$ human activity recognition.

Towards this end, as the softmax layer aims to distinguish the training samples' feature representations belonging to different classes, since feature representation especially of the negative class may extremely expanded in the feature space generated by the neural layer, we propose to attach a regularization layer that aims at shrinking the radius of the MEB of the training samples.

Let us denote by $\mathcal{X} = \{\mathbf{X}_i, i = 1, \ldots, N\}$ the set of N training images, and by $\mathcal{Y}^L = \{\mathbf{y}_i^L, i = 1, \ldots, N\}$ the set of N feature representations of the deep neural layer, L. We abbreviate as R_{MEB}, the radius of the minimum enclosing ball of all the training samples. The squared radius is formally expressed by the following equation:

$$R_{MEB}{}^2 = \min_{R, \boldsymbol{y}_0^L} R^2, \quad s.t. \quad \|\boldsymbol{y}_i^L - \boldsymbol{y}_0^L\|_2^2 \leq R^2, \quad \forall i, \tag{13}$$

where $\boldsymbol{y}_0^L$ is the centroid of all the training samples $\boldsymbol{y}_i^L$.

However, this definition suffers from a major shortcoming. That is, it can not be applied in terms of mini-batch training, since it requires the centroid of all the training data. In order to tackle this issue, we utilize an approximation of the above definition. We express the radius of the minimum enclosing ball of the training data, using the maximum pairwise distance over all pairs of training samples. That is:

$$\tilde{R}_{MEB}^2 = \max_{i,j} \|\boldsymbol{y}_i^L - \boldsymbol{y}_j^L\|_2^2 \tag{14}$$

In [46] the authors proved that the radius R_{MEB} is well approximated by $\tilde{R}_{MEB}$ with the following inequality:

$$\tilde{R}_{MEB} \leqslant R_{MEB} \leqslant \frac{1+\sqrt{3}}{2} \tilde{R}_{MEB} \tag{15}$$

Thus, instead of minimizing the squared radius of the smallest sphere enclosing all the training samples, for simplicity we minimize the squared diameter that is defined by the maximum pairwise distance over all pairs of the training samples, since this does not affect the solution of the minimization problem, and following also the work in [46].

Subsequently, since the approximated radius is defined over all the pairs of training samples, we first formulate the following minimization problem utilizing the softmax function over the max operator which is non-smooth, as it is shown in [48] and then we further relax the approximated radius to make it appropriate for mini-batch training:

$$\min_{\mathbf{y}_i^L \in \mathcal{Y}^L} \mathcal{J}_{MEB} = \min_{\mathbf{y}_i^L \in \mathcal{Y}^L} \sum_{i,j}^{N} k_{ij} \|\mathbf{y}_i^L - \mathbf{y}_j^L\|_2^2, \tag{16}$$

where

$$k_{ij} = \frac{e^{a\|\mathbf{y}_i^L - \mathbf{y}_j^L\|_2^2}}{\sum_{i,j}^{N} e^{a\|\mathbf{y}_i^L - \mathbf{y}_j^L\|_2^2}} \tag{17}$$

measures the correlation of the two samples, while the parameter a controls the approximation degree to max operator. When a is infinite, the approximation is identical to the max operator, while when $a = 0$, $k_{ij} = \frac{1}{N^2}$. The relaxed definition of eq. (16) allows for defining the minimization objective in terms of mini-batch training, instead of the whole dataset. That is, for a set $\mathcal{B}$ of training samples' feature representations of a batch, eq. (16) becomes:

$$\min_{\mathbf{y}_i^L \in \mathcal{Y}^L} \mathcal{J}_{MEB} = \min_{\mathbf{y}_i^L \in \mathcal{Y}^L} \sum_{\mathbf{y}_i^L, \mathbf{y}_j^L \in \mathcal{B}} k_{ij} \|\mathbf{y}_i^L - \mathbf{y}_j^L\|_2^2, \tag{18}$$

Thus, for $a = 0$, $k_{ij} = \frac{1}{|\mathcal{B}|^2}$, where $|\mathcal{B}|$ is the cardinality of set $\mathcal{B}$, the minimization problem can be formulated as follows in terms of mini batch training:

$$\min_{\mathbf{y}_i^L \in \mathcal{Y}^L} \mathcal{J}_{MEB} = \min_{\mathbf{y}_i^L \in \mathcal{Y}^L} \sum_{\mathbf{y}_i^L \in \mathcal{B}} \|\mathbf{y}_i^L - \boldsymbol{\mu}\|_2^2, \tag{19}$$

where $\boldsymbol{\mu} = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{y}_j^L \in \mathcal{B}} \mathbf{y}_j^L$.
The analysis of the above equality can be found in A.

Hence, the total loss is formulated similarly to (12) and the optimization problem is solved using gradient descent.

We note that Support Vector Data Description method, [49], inspired by the Support Vector Classifier, proposes a MEB-like objective in the One Class Classification problem, as the main objective in order to find the outliers. However, the proposed regularizer, as mentioned previously, is rooted in the radius-margin based Support Vector Machines (SVM) [44], [45], [46], and proposes the MEB objective as a regularization in the main classification objective, in order to improve the generalization ability of the binary classifier.

3.3.3 Graph Embedded Regularization Framework

As we have already stated, different regularizers, motivated by different GE approaches, can be defined, apart from the DA and MEB. That is, the proposed regularization method is straightforward, and can materialize all the GE algorithms, and thus it can be seen as a generic graph-embedded regularization framework. In this subsection, we define two additional regularizers inspired by the Locally Linear Embedding (LLE) algorithm, [50], as well as by Clustering-based Discriminant Analysis (CDA), [51].

LLE-inspired regularization: Motivated by the LLE algorithm, which maps the input data to a lower dimensional space in a manner that preserves the relationship between the neighboring samples, a corresponding regularizer can be implemented. That is, we consider a set $\mathcal{X} = \{\mathbf{X}_i, i = 1, \ldots, N\}$ of N images of the training set, and a set $\mathcal{Y}^L = \{\mathbf{y}_i^L, i = 1, \ldots, N\}$ of their corresponding feature representations emerged in the L layer of a deep neural model. For each feature representation $\boldsymbol{y}_i^L$, we consider the set $\mathcal{N}_{(i)}$ that contains its nearest neighbors. Then our goal is to minimize the Euclidean distance between each feature representation $\boldsymbol{y}_i^L$ and the mean vector $\boldsymbol{\mu}_{nn}^i$ of its nearest representations.

Thus, the following optimization problem is defined:

$$\min_{\mathbf{y}_i^L \in \boldsymbol{\mathcal{Y}}^{\mathcal{L}}} \mathcal{J}_{LLE} = \min_{\mathbf{y}_i^L \in \boldsymbol{\mathcal{Y}}^{\mathcal{L}}} \sum_{i=1}^{N} \|\mathbf{y}_i^L - \boldsymbol{\mu}_{nn}^i\|_2^2, \tag{20}$$

where $\boldsymbol{\mu}_{nn}^i = \frac{1}{|N_{(i)}|} \sum_{\mathbf{y}_j^L \in N_{(i)}} \mathbf{y}_j^L$, and $|N_{(i)}|$ denotes the cardinality of set $\mathcal{N}_{(i)}$.
However, we should note that finding the nearest representations for each training sample, comes with additional computational cost.

Clustering-based DA regularization: The problem of crowd detection is a two-class problem of crowded and non-crowded scenes. Thus, in the proposed DA regularizer we consider one centroid for crowded and one for non-crowded scenes and the proposed regularizer brings each training sample closer to the corresponding centroid. However, the nature of the problem, that is the one class describes crowded scenes, while the other class describes anything than crowd, motivates us to explore the number of centroids, aiming at exploiting the extremely wide variation of non crowded scenes, but also variations in crowded scenes (e.g. crowd density, distance, viewpoint etc.). Thus, inspired by the CDA, which assumes a multimodal data distribution inside classes where each class consists of several subclasses, and aims to enhance the between class discrimination by minimizing the scatter within each subclass, while separating subclasses belonging to different classes, we propose the CDA regularizer.

That is, let $\mathcal{X} = \{\mathbf{X}_i, i = 1, \ldots, N\}$ be the set of N images of the training set, $\boldsymbol{\mathcal{Y}}^L = \{\mathbf{y}_i^L, i = 1, \ldots, N\}$ the set of N feature representations emerged in the L layer of a deep neural model. Let also $\mathcal{S}^{ck}$ be the set of all image representations $\boldsymbol{y}_i^L$, that belong to the k-th subclass of the c-th class.

Then, the following optimization problem is defined:

$$\min_{\mathbf{y}_i^L \in \boldsymbol{\mathcal{Y}}^{\mathcal{L}}} \mathcal{J}_{CDA} = \min_{\mathbf{y}_i^L \in \boldsymbol{\mathcal{Y}}^{\mathcal{L}}} \sum_{i=1}^{N} \|\mathbf{y}_i^L - \boldsymbol{\mu}_{ck}^i\|_2^2, \tag{21}$$

where $\boldsymbol{\mu}_{ck} = \frac{1}{|\mathcal{S}^{ck}|} \sum_{\mathbf{y}_j^L \in \mathcal{S}^{ck}} \mathbf{y}_j^L$, $|\mathcal{S}^{ck}|$ is the cardinality of set $\mathcal{S}^{ck}$, and $\boldsymbol{\mu}_{ck}^i = \boldsymbol{\mu}_{ck}$, if $\boldsymbol{y}_i^L \in \mathcal{S}^{ck}$.
However, this approach also comes with additional computational cost of using a clustering algorithm in order to define subclasses in each class.

4 Crowd-Drone Dataset Construction

We have constructed a new *Crowd-Drone* dataset, in order to evaluate the performance of the proposed method, since there is no publicly available crowd dataset of drone-captured videos and images. That is, we created a new dataset by querying specific keywords to the Youtube[4] video search engine. More specifically, we collected 57 drone videos using keywords that describe crowded events (*e.g.* marathon, festival, parade, political rally, protests, etc). We also selected non-crowded videos by searching for unspecified drone videos. Non-crowded images (e.g. cars, buildings, bikes, etc.) also randomly gathered from the senseFly-Example-drone[5], as well as the UAV123[6] datasets. Sample frames from the gathered video sequences are provided in Fig.1. Subsequently, we manually annotated crowded regions from the extracted frames. A total number of 5,920 crowded regions and an equal number of non-crowded images formulated the *Crowd-Drone* dataset. Sample annotated regions are provided in Fig.2. The following pre-processing steps applied to the formed dataset: channel swapping to BGR, rescaling to $[0, 255]$ and mean substraction.

Fig. 1: Sample frames of the selected videos.

Fig. 2: Sample regions of the *Crowd-Drone* dataset.

5 Experiments

In this section, we present the experiments performed in order to evaluate the proposed method. Throughout this work, we use Test Accuracy to evaluate the proposed methods for the crowd detection problem. Qualitative results are also provided through the crowd heatmaps. The proposed CNN model serves as baseline for the proposed regularization approaches. We also compare the proposed approaches with the L1 and L2 regularization schemes. In the following, we first describe the utilized CNN architecture, then we report the implementation details of the proposed method, and finally we present the evaluation results.
The proposed Crowd Detector as well as the links of the utilized drone videos are available at: https://github.com/mtzelepi/GraphEmbeddedCNN.

5.1 CNN Model

The proposed CNN model contains six learned convolutional layers. The network accepts RGB images of size $128 \times 128 \times 3$. The output of the last convolutional layer is fed to a Softmax layer which produces a distribution over the 2 classes of crowd and non-crowd. Each convolutional layer except for the last one is followed by a Parametric Rectified Linear Unit (PReLU) activation layer which learns the parameters of the rectifiers, since it has been proven to enhance the classification results [52]. Max-pooling layers follow the first and the fifth convolutional

4. http://www.youtube.com/
5. https://www.sensefly.com/drones/example-datasets.html
6. https://ivul.kaust.edu.sa/Pages/Dataset-UAV123.aspx

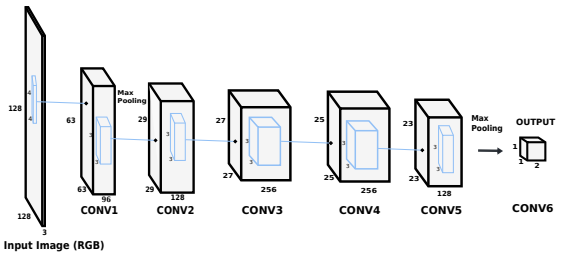


Fig. 3: Overview of the proposed CNN architecture

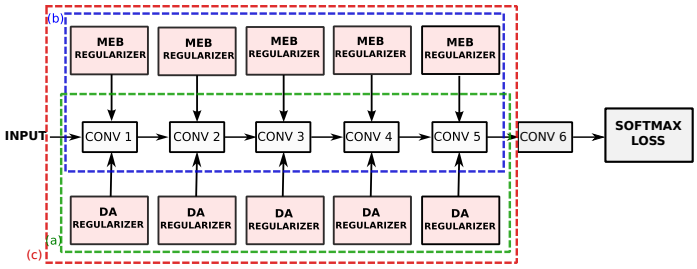


Fig. 4: In (a) the DA regularizer is attached to the deep neural layers of the proposed CNN model. In (b) the MEB regularizer is correspondingly attached to all the neural layers, while in (c) both the proposed regularizers are attached to the network layers. In each of the three cases the regularizers can be attached on not only all the layers but on individual ones as well as on combinations of layers.

layers, aiming to reduce the spatial size of their input and the number of the parameters contributing also to the overfitting control, while a response-normalization layer is utilized after the first pooling layer, in order to aid generalization. A Dropout layer [53] with probability 0.5 follows the fifth convolutional layer aiming to reduce overfitting. An overview of the proposed model is illustrated in Figure 3.

5.2 Implementation Details

The proposed CNN model was implemented and trained using the Caffe Deep Learning framework [54]. We use the mini-batch gradient descent for the network training. That is, an update is performed for every mini-batch of N_b training samples. The learning rate is set to 10^{-5}, and the batch size is set to 64. The weight decay is 0.0005, and the momentum is 0.9. All the models are trained on an NVIDIA GeForce GTX 1080 with 8GB of GPU memory.

As mentioned before, the proposed regularizers can be applied on all the neural layers, on individual layers, as well as on multiple layers. Additionally, both the proposed regularizers can be applied on a certain layer. The idea behind this approach is that the regularizers attempt to simultaneously shrink the training samples, and bring closer samples of the same class, while the softmax layer aims at separating the samples of different classes. In our experiments, apart from the case we apply the regularizers on all the convolutional layers, we also examine to apply them on the CONV5, on both the CONV4 and CONV5, on CONV3, CONV4 and CONV5, and finally on CONV2, CONV3, CONV4 and CONV5. We also apply both the DA and the MEB regularizers on all the convolutional layers. The above sets of experiments are illustrated in Fig. 4. To do this, we attach

Training Approach	Test Accuracy
Softmax	0.9405 ± 0.0079
Softmax & L1	0.9435 ± 0.009
Softmax & L2	0.9422 ± 0.005
Softmax & MEB	0.9541 ± 0.0072
Softmax & DA	0.9546 ± 0.0061

TABLE 1: Crowd Dataset: Test Accuracy

an additional pooling layer on each of these layers, the so-called Maximum Activations of Convolutions (MAC) [39] layer that implements the max-pooling operation over the width and height of the output volume, for each of the 128 feature maps of the CONV5 layer, correspondingly of the 256 feature maps of the CONV4 and CONV3 layers, and so on. The MAC layer on CONV5 outputs a 128-d vector, while the MAC layer on CONV4 and CONV3 layers outputs a 256-d vector representation of each input image. Subsequently, according to the regularization approach we formulate the desired targets, and we attach the Euclidean loss layer.

The Euclidean loss is initially significantly larger than the softmax one. Thus, in order to control the relative importance of the contributed losses, we first set the Euclidean loss parameter η in (10) to 0.0001, and we fixed it to 0.01 at the 20 epochs up to the final epoch, for all the convolutional layers. All the models are trained for 100 epochs.

5.3 Experimental Results

In order to validate the performance of the proposed regularizers, we use the 5-fold cross-validation, and we report the mean Test Accuracy of the five cases and the standard deviation. We note that the five folds are non-overlapped and fixed for all the compared methods. In Table 1 we present the performance of the proposed regularizers against the softmax-only approach, in terms of classification accuracy. We also compare the regularizers with the standard L1 and L2 regularization schemes. From the demonstrated results, we can see that both the MEB and DA regularizers improve the classification performance, while they are also superior over the L1 and L2 regularizers which slightly improve the results.

Subsequently, we use the t-distributed stochastic neighbor embedding (t-SNE) [55] algorithm, a non-parametric technique for dimensionality reduction, widely used for data visualization, to visualize the 128-d feature representations generated by the CONV5 layer of the proposed MEB and DA models, as well as of the baseline Softmax model, for 100 crowded and 100 non-crowded images. Thus, in 5a, 5b, and 5c of Fig. 5 we illustrate the 2-d t-SNE embedding of the CONV5 representations at 1 epoch, the t-SNE embedding of the representations at 10 epochs, and at 20 epochs of Softmax training, respectively. In 5d, 5e, and 5f of Fig. 5 we illustrate the corresponding representations of the MEB training, and in 5g, 5h, and 5i of the same figure we provide the corresponding DA representations. As we can observe, while the main Softmax Loss aims at separating the samples of different classes, the DA regularizer seeks to bring the training samples' representations of the same class together, and the MEB regularizer aims to shrink the radius of the MEB of the training samples' representations. Both the regularizers induce the training samples' representations to shrink, while the DA regularizer also preserves discriminant power. That is, if we push them to their extreme, the training samples would

Training Approach	Test Accuracy
Hinge Loss	0.9488 ± 0.0009
Hinge Loss & MEB	0.9541 ± 0.0022
Hinge Loss & DA	0.9584 ± 0.003

TABLE 2: Crowd Dataset - Hinge Loss: Test Accuracy

collapse to one point for the MEB regularizer, and two points for the DA regularizer (possibly even to one too).
In Fig. 6 and Fig. 7 we compare the proposed regularization techniques, applied on all the convolutional layers, against the one-loss training of the proposed CNN architecture. From the demonstrated results, several remarks can be drawn. First, we can notice that each of the proposed approaches accomplishes improved results in terms of accuracy as compared to the one-loss model. Second, we observe that the regularizers cause weak improvement up to 20 epochs, which is reasonable since the weight parameter η in (10) for the DA regularizer and similarly for the MEB one, is intentionally lower to the first epochs, as stated previously. In Fig. 9 for the DA regularizer, as well as in Fig. 10 for the MEB regularizer, it is shown that the number of the regularizers attached to the deep neural layers affects the classification accuracy. That is, the more layers we attach the regularizer the better improvement we accomplish, as it is clearly illustrated in the Figures. This also comes with impact on the computational cost. Finally, in Fig. 8 we apply both the DA and MEB regularizers, and we see the improvement against the one-loss model.

Furthermore, as we have previously mentioned, the Hinge Loss layer could also utilized for the classification task, instead of the Softmax Loss layer. To this aim, we also perform experiments on the Crowd-Drone dataset utilizing the Hinge Loss instead of the Softmax one, and we apply the proposed MEB regularizer, as well as the DA regularizer. As we can observe in Table 2 we can indeed achieve improved results with the proposed regularizers using the Hinge Loss as the classification objective.

Subsequently, in order to validate our claim that the proposed regularization scheme can be applied for generic classification purposes, we also conducted additional experiments on two additional datasets. We utilized two datasets, constructed for Bicycle (i.e. bicycle with bicyclist) and Football Player Detection, in the context of media coverage of certain sport events (e.g. football match, bicycle race) by multiple drones with increased decisional autonomy. The first dataset, namely Bicycle Dataset, consists of 59,904 train images of bicycles and non-bicycles, and correspondingly of 7,000 test images. The input images are of size 64×64, and thus we modified the proposed architecture for the crowd detection by removing the pooling layer which follows the 5^{th} convolutional layer. The second dataset, namely Football Dataset, consists of 51,200 images of football players and non football players. The input images are of size 32×32, and hence, except for the aforementioned pooling layer, we have also removed the first pooling layer in the proposed Crowd architecture. In Table 3 we provide the evaluation results in terms of classification accuracy, for the Bicycles Dataset. We repeated the experiments five times, and we report the mean value and the standard deviation. The batch size is set to 256. As we can see the proposed regularizers significantly improve the performance of the softmax-only model. We also compare the proposed regularizers with the L1 and L2 regularization schemes. As we can observe the L1 regularization impairs the classification performance, while the L2 one marginally improves the performance.

Training Approach	Test Accuracy
Softmax	0.9119 ± 0.004
Softmax & L1	0.8991 ± 0.0079
Softmax & L2	0.9134 ± 0.0021
Softmax & MEB	0.9448 ± 0.0057
Softmax & DA	0.9423 ± 0.0045

TABLE 3: Bicycle Dataset: Test Accuracy

In Table 4 we provide the evaluation results in terms of classification accuracy, for the Football Dataset. We repeated the experiments five times, and we report the mean value and the standard deviation. The batch size is set to 256. We compare the proposed regularizers with the softmax-only approach as well as the L1 and L2 regularizers. As we can see the proposed regularizers are notably superior over both the softmax-only and the L1/L2 regularization approaches.

Training Approach	Test Accuracy
Softmax	0.8850 ± 0.0051
Softmax & L1	0.8834 ± 0.005
Softmax & L2	0.8856 ± 0.0083
Softmax & MEB	0.9112 ± 0.01
Softmax & DA	0.9128 ± 0.003

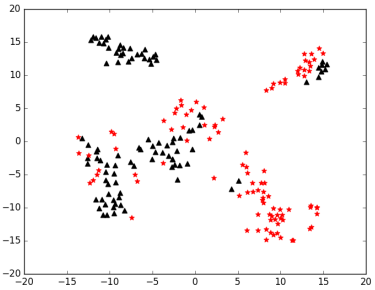
TABLE 4: Football Dataset: Test Accuracy

Thereafter, for ranking the proposed regularization methods and evaluating the statistical significance of the obtained results, we performed a post-hoc Bonferroni test [56]. The performance of pairwise methods is significantly different, if the corresponding mean ranks over the datasets differ by at least the critical difference which is defined as:

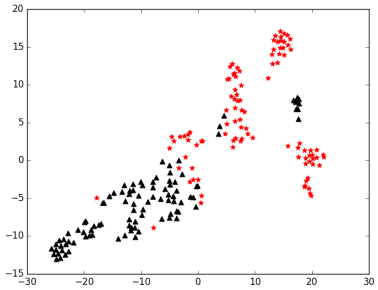
$$CD = q_a \sqrt{\frac{m(m+1)}{6D}}, \tag{22}$$

where m corresponds to the number of the compared methods, D to the number of datasets, and critical values q_α can be found in [56]. In the performed comparisons we set $\alpha = 0.05$. In the experiment we considered the five folds of the Crowd Dataset as five discrete datasets, since the folds are non-overlapped and fixed for all the compared methods. Thus, the number of datasets is seven in the performed test, including the Bicycle and Football datasets. The compared methods are five, that is the proposed regularizers, as well as the L1 and L2 regularizers are compared with a control method which is the softmax-only approach. The ranking results are also illustrated in Fig. 11. On the vertical axis the five compared methods are depicted, while on the horizontal axis is depicted the performance ranking. The circles denote the mean rank, while the intervals around them denote the confidence interval, as it is defined by the CD value. Non-overlapping intervals between two compared methods indicate that the methods are significantly different, while overlapping intervals imply that there is not a statistically significant difference between the corresponding ranks. As we can observe, both the proposed regularizers are significantly different against the softmax-only training, while between L1/L2 regularization and the softmax-only approach there is not significant difference.

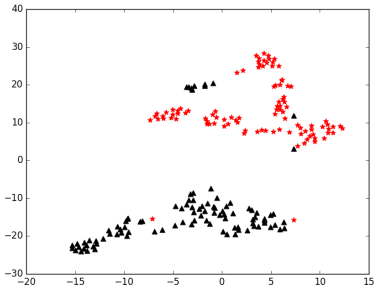
Qualitative results: In Fig. 12 we provide the heatmaps for the class Crowd of the proposed DA-regularized classification



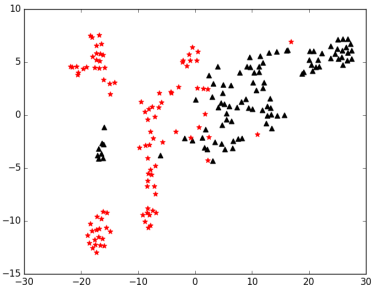
(a) Representations at 1 epoch of Softmax training



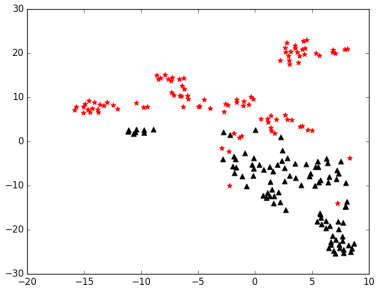
(b) Representations at 10 epochs of Softmax training



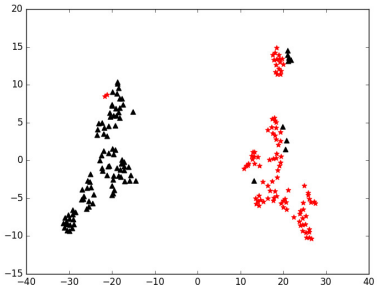
(c) Representations at 20 epochs of Softmax training



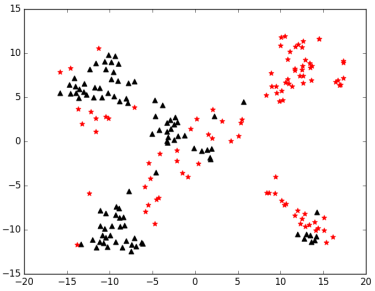
(d) Representations at 1 epoch of MEB training



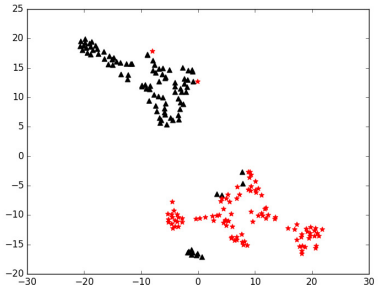
(e) Representations at 10 epochs of MEB training



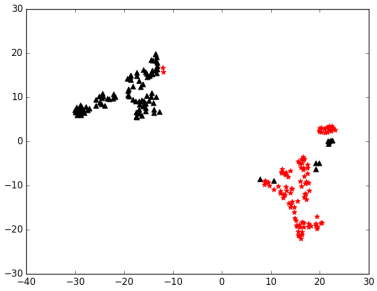
(f) Representations at 20 epochs of MEB training



(g) Representations at 1 epoch of DA training



(h) Representations at 10 epochs of DA training



(i) Representations at 20 epochs of DA training

Fig. 5: Visualization by t-SNE

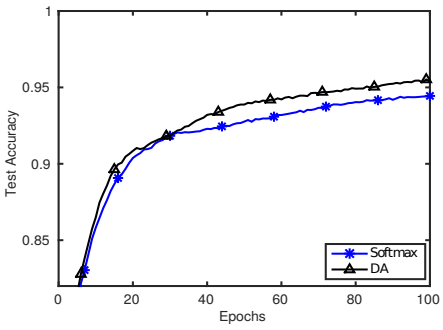


Fig. 6: DA Regularizer

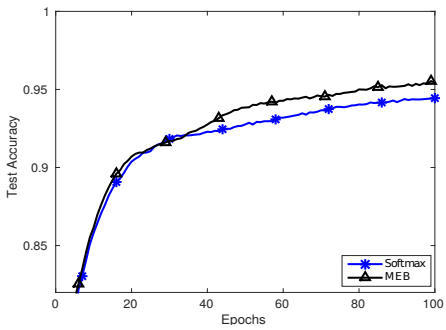


Fig. 7: MEB Regularizer

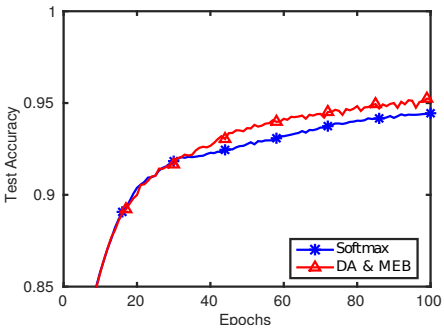


Fig. 8: Both DA and MEB Regularizer

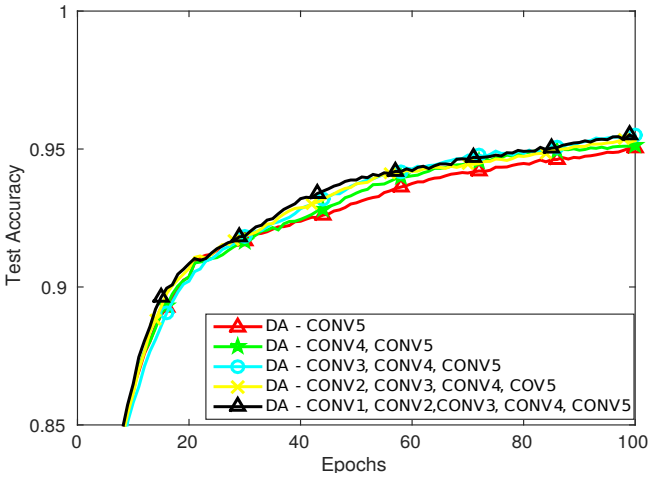


Fig. 9: DA Regularizer - Comparison among utilized layers

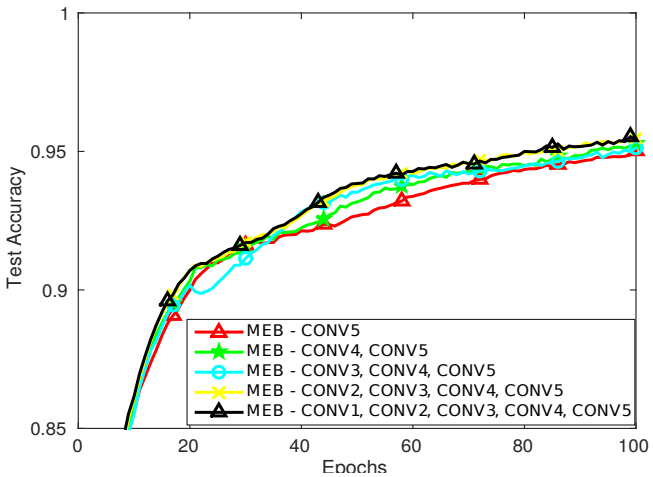


Fig. 10: MEB Regularizer - Comparison among utilized layers

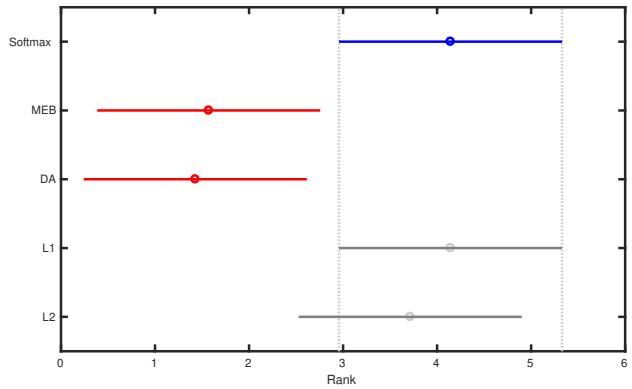


Fig. 11: Post-Hoc Bonferroni Test

model, trained on the entire dataset of 11,840 images, since the test frames are not included in the dataset. That is, unseen images of size 1024×1024 are fed to the network, and for every window 128×128 with stride 8, we compute the output of the network at the layer CONV6, for the label Crowd, which is the desired heatmap. We should note that proposed method can obtain pixel-level heatmaps, albeit it uses data labelled at image level in the training procedure. That is, it produces a heatmap as output, similarly to more complex methods [57] although it is not trained in pixel level segmented images. More heatmaps of crowded and totally non-crowded scenes are provided as supplementary material.

5.3.1 Discussion on Speed

We tested the proposed crowd detector on a GeForce GTX 1080 GPU for various input sizes, and we compare it in terms of frames per second (fps) with a common baseline model (i.e. VGG-16 [40]) for the latter's fixed input. Since the deployment of the crowd detector will be done on a drone, we also tested the performance on a low-power NVIDIA Jetson TX2 module with 8GB of memory, which is a state of the art GPU used for on-board drone perception. The experimental results are presented in Table 5. As we can see, the proposed model operates at 49.7 fps for input of size 224×224, against the baseline model which runs at 9.36 fps for the same fixed input on the Jetson TX2 module, whereas it runs at 13.1 fps for input of size 512×512, and at 2.1 fps for input of size 1024×1024.

Generally, utilizing a CNN pretrained model as a weight initialization constitutes a common practice in deep learning, in order to tackle the issue of limited data, or as a way to boost the performance of a model. A common pretrained CNN model like VGG, requires fixed input due to the fully connected layers. However, this is prohibitive for the specific application, since handling images with arbitrary dimension is required in order to estimate a heatmap of the crowded areas. This can be obtained using only convolutional layers. A way to tackle the issue of the fully convolutional requirement of the application, is to adapt the VGG pretrained model by removing the fully connected layers. However, even if we discard the fully connected layers of the model, the application imposes a lightweight model, capable of being applied on-board. For example, the modified fully convolutional VGG model runs at 28.16 fps for input 512×512 on the GTX 1080 (against 99.4 fps of the proposed one), while for an input of size 1024×1024 it is out of memory even in the GTX 1080 (the proposed model runs at 23.45 fps, respectively). We should finally highlight that even if it is feasible to apply such an architecture for the task of crowd detection, this choice remains prohibitive, since the crowd detector is one of a series of models that should be deployed and run simultaneously on-board (e.g. bicycle/bicyclist detection, football player detection, pose estimation, etc.) in the context of media coverage of certain sport events by drones.

Model	Input	Jetson TX2	GeForce GTX 1080
VGG	224×224	9.36	89.52
Proposed	224×224	49.7	416.66
Proposed	512×512	13.1	99.4
Proposed	1024×1024	2.1	23.45

TABLE 5: Speed (FPS)

5.3.2 Training with limited data

Finally, we investigate the scenario of training with limited data. That is, we conducted an additional experiment in order to evaluate the performance of the proposed regularizers under the extreme case where the available training samples are very limited, and hence the utilization of a pretrained CNN model is

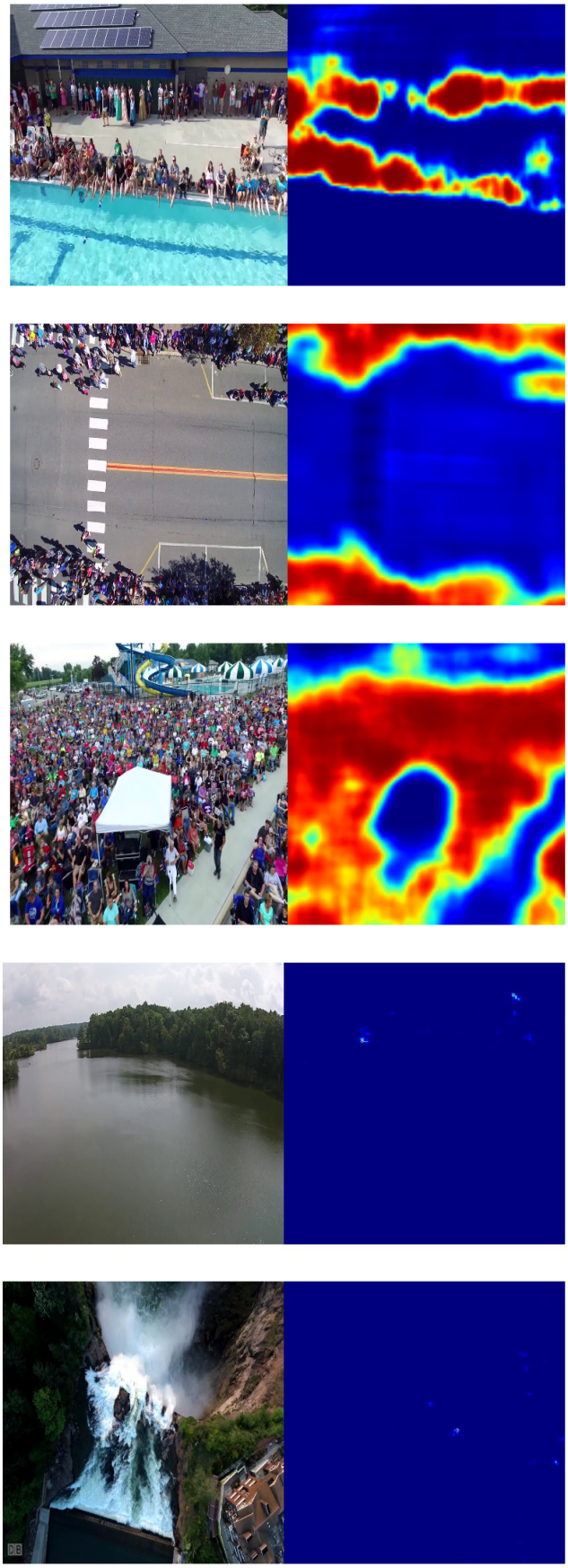


Fig. 12: Initial images on the left and the corresponding detected crowd heatmaps on the right for five test images

mandatory. More specifically, we have considered as training set only 2,048 out of the 11,840 images of the Crowd dataset (using 1,024 images of crowd and 1,024 images of non-crowd), and as test set 7,000 images. We used the fully convolutional portion of the VGG-16 model, and we added a convolutional layer with kernels covering the entire input. The output of the additional convolutional layer was fed to a Softmax layer, producing a distribution over the 2 classes of Crowd and Non-Crowd. We trained an one-loss model using as weight initialization the modified fully convolutional VGG model. Subsequently, we also attached the two proposed regularizers. We note, that both in the case of the softmax-only training as well as in the two-loss regularized training, we only trained the two last convolutional layers. The experimental results are provided in Table 6. As we can observe, the VGG initialization indeed helps the model, trained with only 2,048 images, to perform well. Additionally, from the demonstrated results it is deduced that the proposed regularizers accomplish improved results as compared to the softmax-only approach, even in the case of the extremely limited data, where we utilize a common CNN pretrained model as weight initialization. That is, the proposed regularization scheme is applicable for generic classification purposes, and for different architectures, while it also enhances the generalization ability of the model, regardless of the size of the training dataset.

Training Approach	Test Accuracy
VGG - Softmax	0.9044 ± 0.003
VGG - Softmax & MEB	0.9136 ± 0.0039
VGG - Softmax & DA	0.9138 ± 0.0035

TABLE 6: Crowd Dataset - Modified-VGG initialization, Limited Training Samples

6 Conclusions

In this paper, a novel human crowd detection method, for drone flight safety purposes, using fully convolutional deep CNNs was proposed. Firstly, a fully convolutional architecture was proposed, in order to satisfy the computational and the memory limitations of our application, and also benefit from the fully convolutional networks properties. Secondly, a novel regularization technique, that borrows ideas from the GE framework, and is applicable to various deep learning models, for generic classification problems, was proposed. Thirdly, a new *Crowd-Drone* dataset was constructed, for the specific task, since there is no publicly available dataset with drone-captured crowded images. The experimental validation on the created dataset proved the effectiveness of both the crowd detector, and the regularizers. Finally, in order to better evaluate the performance of the proposed regularizers, as well as to validate our claim that the proposed regularization scheme is applicable in generic classification problems, additional experiments were also conducted on two new datasets, where the enhanced performance of the regularizers was also validated.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information

it contains. We would like to thank RAI (Radiotelevisione italiana S.p.A.) for providing videos of Giro d' Italia, and Danai Triantafyllidoy for providing the annotations for the Football and Bicycle datasets.

Appendix

In this appendix we provide the analysis of equality 19. For simplicity in notation we denote $\mathbf{y}^L$ as $\mathbf{y}$.

$$\sum_{\mathbf{y}_i,\mathbf{y}_j\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}\|\mathbf{y}_i-\mathbf{y}_j\|_2^2 = \sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}(\mathbf{y}_i-\mathbf{y}_j)^T(\mathbf{y}_i-\mathbf{y}_j) = \sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}\mathbf{y}_i^T\mathbf{y}_i - \sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}\mathbf{y}_i^T\mathbf{y}_j - \sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}\mathbf{y}_j^T\mathbf{y}_i + \sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}\mathbf{y}_j^T\mathbf{y}_j = \tag{23}$$

$$|\mathcal{B}|\sum_{\mathbf{y}_i\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}\mathbf{y}_i^T\mathbf{y}_i + |\mathcal{B}|\sum_{\mathbf{y}_j\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}\mathbf{y}_j^T\mathbf{y}_j - 2\sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}} \frac{1}{|\mathcal{B}|^2}\mathbf{y}_i^T\mathbf{y}_j = \frac{2}{|\mathcal{B}|}\Big(\sum_{\mathbf{y}_i\in\mathcal{B}}\mathbf{y}_i^T\mathbf{y}_i - \frac{1}{|\mathcal{B}|}\sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}}\mathbf{y}_i^T\mathbf{y}_j\Big)$$

Also,

$$\sum_{\mathbf{y}_i\in\mathcal{B}}\|\mathbf{y}_i-\boldsymbol{\mu}\|_2^2 = \sum_{\mathbf{y}_i\in\mathcal{B}}(\mathbf{y}_i-\frac{1}{|\mathcal{B}|}\sum_{\mathbf{y}_j\in\mathcal{B}}\mathbf{y}_j)^T(\mathbf{y}_i-\frac{1}{|\mathcal{B}|}\sum_{\mathbf{y}_k\in\mathcal{B}}\mathbf{y}_k) = \sum_{\mathbf{y}_i\in\mathcal{B}}(\mathbf{y}_i^T\mathbf{y}_i - \frac{\mathbf{y}_i^T}{|\mathcal{B}|}\sum_{\mathbf{y}_k\in\mathcal{B}}\mathbf{y}_k - \frac{1}{|\mathcal{B}|}\sum_{\mathbf{y}_j\in\mathcal{B}}\mathbf{y}_j^T\mathbf{y}_i + \frac{1}{|\mathcal{B}|^2}\sum_{\mathbf{y}_j\in\mathcal{B}}\mathbf{y}_j^T\sum_{\mathbf{y}_k\in\mathcal{B}}\mathbf{y}_k) = \sum_{\mathbf{y}_i\in\mathcal{B}}\mathbf{y}_i^T\mathbf{y}_i - \frac{2}{|\mathcal{B}|}\sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}}\mathbf{y}_i^T\mathbf{y}_j + \frac{|\mathcal{B}|}{|\mathcal{B}|^2}\sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}}\mathbf{y}_i^T\mathbf{y}_j = \sum_{\mathbf{y}_i\in\mathcal{B}}\mathbf{y}_i^T\mathbf{y}_i - \frac{1}{|\mathcal{B}|}\sum_{\mathbf{y}_i\in\mathcal{B}}\sum_{\mathbf{y}_j\in\mathcal{B}}\mathbf{y}_i^T\mathbf{y}_j \tag{24}$$

For simplicity, $\frac{2}{|\mathcal{B}|}$ can be omitted, since it does not affect the solution of the minimization problem.

References

[1] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems 2*. Morgan Kaufmann Publishers Inc., 1990, pp. 396–404.

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

[3] 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.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[7] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[8] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 1701–1708.

[9] M. Tzelepi and A. Tefas, "Deep convolutional learning for content based image retrieval," *Neurocomputing*, vol. 275, pp. 2467–2478, 2018.

[10] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[11] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective*, vol. 261, p. 276, 1995.

[12] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1653–1660.

[13] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 3626–3633.

[14] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.

[15] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 40–51, 2007.

[16] A. Iosifidis, A. Tefas, and I. Pitas, "Graph embedded extreme learning machine," *IEEE transactions on cybernetics*, vol. 46, no. 1, pp. 311–324, 2016.

[17] D. Bouzas, N. Arvanitopoulos, and A. Tefas, "Graph embedded non-parametric mutual information for supervised dimensionality reduction," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 5, pp. 951–963, 2015.

[18] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[19] J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1168–1175.

[20] C. Wu, P. Karanasou, M. J. Gales, and K. C. Sim, "Stimulated deep neural network for speech recognition," University of Cambridge Cambridge, Tech. Rep., 2016.

[21] C. Wu, M. J. Gales, A. Ragni, P. Karanasou, and K. C. Sim, "Improving interpretability and regularization in deep learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 256–265, 2018.

[22] M. Tzelepi and A. Tefas, "Human crowd detection for drone flight safety using convolutional neural networks," in *European Signal Processing Conference (EUSIPCO), Kos, Greece, 2017*.

[23] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, "Recent survey on crowd density estimation and counting for visual surveillance," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 103–114, 2015.

[24] H. Fradi and J.-L. Dugelay, "Spatial and temporal variations of feature tracks for crowd behavior analysis," *Journal on Multimodal User Interfaces*, vol. 10, no. 4, pp. 307–317, 2016.

[25] P. Reisman, O. Mano, S. Avidan, and A. Shashua, "Crowd detection in video sequences," in *Intelligent Vehicles Symposium, 2004 IEEE*. IEEE, 2004, pp. 66–71.

[26] O. Arandjelovic, "Crowd detection from still images," in *BMVC 2008: Proceedings of the British machine vision association conference 2008*. BMVA Press, 2008, pp. 1–10.

[27] O. Meynberg, S. Cui, and P. Reinartz, "Detection of high-density crowds in aerial images using texture classification," *Remote Sensing*, vol. 8, no. 6, p. 470, 2016.

[28] L. Boominathan, S. S. Kruthiventi, and R. V. Babu, "Crowdnet: a deep convolutional network for dense crowd counting," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 640–644.

[29] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 833–841.

[30] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 589–597.

[31] J. Shao, K. Kang, C. Change Loy, and X. Wang, "Deeply learned attributes for crowded scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4657–4666.

[32] M. Ravanbakhsh, M. Nabi, H. Mousavi, E. Sangineto, and N. Sebe, "Plug-and-play cnn for crowd motion analysis: An application in abnormal event detection," *arXiv preprint arXiv:1610.00307*, 2016.

[33] D. Babu Sam, S. Surya, and R. Venkatesh Babu, "Switching convolutional neural network for crowd counting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5744–5752.

[34] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2547–2554.

[35] H. Idrees, K. Soomro, and M. Shah, "Detecting humans in dense crowds using locally-consistent scale prior and global occlusion reasoning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 1986–1998, 2015.

[36] A. Dehghan and M. Shah, "Binary quadratic programing for online tracking of hundreds of people in extremely crowded scenes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 568–581, 2018.

[37] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," *arXiv preprint arXiv:1801.04264*, 2018.

[38] T. Castelli, A. Sharghi, D. Harper, A. Tremeau, and M. Shah, "Autonomous navigation for low-altitude uavs in urban areas," *arXiv preprint arXiv:1602.08141*, 2016.

[39] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," *CoRR*, vol. abs/1511.05879, 2015.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[41] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal component analysis*. Springer, 1986, pp. 115–128.

[42] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[44] H. Do, A. Kalousis, and M. Hilario, "Feature weighting using margin and radius based error bound optimization in svms," *Machine Learning and Knowledge Discovery in Databases*, pp. 315–329, 2009.

[45] H. Do, A. Kalousis, A. Woznica, and M. Hilario, "Margin and radius based multiple kernel learning," *Machine Learning and Knowledge Discovery in Databases*, pp. 330–343, 2009.

[46] H. Do and A. Kalousis, "Convex formulations of radius-margin based support vector machines," in *International Conference on Machine Learning*, 2013, pp. 169–177.

[47] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. Wiley New York, 1998, vol. 1.

[48] L. Lin, K. Wang, W. Zuo, M. Wang, J. Luo, and L. Zhang, "A deep structured model with radius-margin bound for 3d human activity recognition," *arXiv preprint arXiv:1512.01642*, 2015.

[49] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[50] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[51] X.-w. Chen and T. Huang, "Facial expression recognition: a clustering-based approach," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1295–1302, 2003.

[52] 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*, 2015, pp. 1026–1034.

[53] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[54] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[55] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[56] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[57] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.