Noname manuscript No.
(will be inserted by the editor)

Class-Specific Discriminant Regularization in real-time deep CNN models for binary classification problems

Maria Tzelepi · Anastasios Tefas

the date of receipt and acceptance should be inserted later

Abstract In this paper, we first propose lightweight deep CNN models, capable of effectively operating in real-time on-drone for high-resolution video input, addressing various binary classification problems, e.g. crowd, face, football player, and bicycle detection, in the context of media coverage of specific sport events by drones with increased decisional autonomy. Furthermore, we propose a novel Class-Specific Discriminant regularizer in order to improve the generalization ability of the proposed real-time models, exploiting the nature of the considered two-class problems. The experimental evaluation on four datasets validates the effectiveness of the proposed regularizer in enhancing the generalization ability of the proposed models.

Keywords Deep Convolutional Neural Networks · Class-Specific Discriminant Regularizer · Real-Time · Lightweight Models · Drones · Binary Classification

1 Introduction

During the recent years deep Convolutional Neural Networks (CNN), [1,2] have been established among the most efficient research directions in a wide spectrum of computer vision tasks, accomplishing superior results over previous shallow algorithms, [3,4,5,6,7]. Apart from developing successful deep models for various computer vision tasks, another research direction that flourishes during the recent few years is the development of lightweight models capable of running on devices with limited computational resources such as mobile phones and embedded systems, [8,9].
Over the recent few years Unmanned Aerial Vehicles, broadly known as *drones*, have

Maria Tzelepi
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
E-mail: mtzelepi@csd.auth.gr

Anastasios Tefas
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
E-mail: tefas@csd.auth.gr

powerfully emerged in a wide range of applications, ranging from entertainment to visual surveillance, medical emergencies [10], and rescue within the context of natural disasters [11]. Their ability to capture spectacular aerial shots or shots of even inaccessible places, gradually displaces previous practices in media production. A key issue associated with the rise of drones is the demand of developing models for various computer vision tasks, capable of addressing the additional challenges of drone-captured images (such as occlusion, unconstrained pose variations, small object size), and also capable of running on-drone, that is with limited processing power.

Thus, in this paper, we propose lightweight deep CNN models allowing real-time deployment for high resolution images for specific classification problems involved in the context of media coverage of certain sport events by multiple drones with increased decisional autonomy. More specifically, we deal with face, bicycle, and football player detection, as well as crowd detection towards crowd avoidance ensuring the drone's safe operation, since a drone may fly in vicinity of crowds, and is potentially exposed to unpredictable errors or environmental hazards that impose emergency landing, and also drone flight regulations in several Countries' national legislation request a safe distance to be maintained between the drone and crowds. Our goal is to provide semantic heatmaps [12], rather than bounding boxes, by predicting for each location within the captured high-resolution scene the object's presence. That is, we train models with RGB input of size either 32×32 or 64×64, and then test images are fed to the network, and for every window 32×32 or 64×64 respectively, we compute the output of the network at the last convolutional layer. An example of a crowd heatmap is provided in Fig. 1. Furthermore, the above procedure finds application in the camera control problem, [13,14], where the goal is to control the camera without using bounding boxes but only visual input. That is, the semantic heatmaps for each of the aforementioned classification problems, aim to assist the algorithm for controlling the camera of the drone for cinematography tasks by sending error signals. We also note that apart from the camera control problem, in some cases the object to be detected (such as crowd) could be distributed in such a way in an image, that it is difficult to be bounded by a box. Thus, in these cases it is more suitable to predict for each patch a probability of crowd existence, and then provide a semantic heatmap of the estimated probability of existence of crowd in each location within the captured scene, instead of a box surrounding the crowded area. We should highlight that it is of utmost importance for the application to handle high resolution images, since objects in drone-captured images are extremely small, and thus image resizing in order to meet real-time deployment limits, that is used by almost all of the state-of-the-art visual content analysis models (e.g. YOLO [15], SSD [16], etc.), would further shrink the object of interest, rendering the detection infeasible. Finally, we should note that even recent drones are capable of providing HD streaming, this comes with a latency, since the drone needs to compress, transmit and decompress the video stream. It was experimentally verified, that a procedure of compressing the drone-captured stream, sending it to the ground station, and decompressing it in order to use it for each of the tasks, has a latency of 120 ms (3 frames). More specifically, the pipeline was as follows: The video stream captured from drone was compressed using H.264 encoding. The compression took place on-drone using the NVIDIA's Jetson TX2 accelerators. Then the compressed stream was transmitted to the ground station via LTE using the RTP protocol, where it was decompressed. Thus, considering a detection/tracking task of fast moving objects, like the ones involved in sport events, the aforementioned latency (120 ms) leads to the loss of the object of interest. Furthermore, the drone

should be able to operate safely, even when the communication to the ground station is infeasible, thus our goal is to develop models for addressing the specific classification problems on-drone, as well as in real-time.

To the best of our knowledge there is no other work in the recent literature proposing real-time models capable of running on devices with limited computational resources for high resolution input. In [17] a computation-efficient CNN model is proposed for mobile devices with limited computing power, and it is shown that in order to achieve real-time deployment, someone has to reduce the input frame resolution to 224×224, sacrificing also the accuracy. However, surveying the relevant literature we can see that several works have emerged towards designing lightweight models. In [9] is proposed to replace 3×3 convolutions with 1×1 convolutions to create a very small network capable of reducing $50\times$ the number of parameters while obtaining high accuracy. In [18] various practical guidelines for efficient network design and a new architecture are proposed. In [19] an efficient convolutional network architecture that allows feature re-use through dense connectivity, and prunes filters associated with superfluous feature re-use through learned group convolutions, is proposed. In [20] the problem of deploying deep neural networks on mobile devices is addressed, and an acceleration method so as to speed up the neural networks with adequate accuracy, by significantly reducing the execution time on non-tensor layers, is proposed.
Subsequently, in [21] the problem of target recognition in synthetic aperture radar images is addressed, proposing a lightweight CNN model based on visual attention mechanism. In [22] a real-time traffic sign recognition system consisting of detection and classification modules is proposed. A color probability model to deal with color information of traffic signs is first proposed, traffic sign proposals are then extracted and, SVM and CNN are combined to detect and classify traffic signs. In [23] the authors deal with dynamic scene classification utilizing two variants of deep convolutional neural networks to encode spatial appearance and short-term dynamics into short-term deep features, and then they propose to extract long-term frequency features using the autoregressive moving average model. Finally, in [24] the authors deal with change detection, proposing a general end-to-end 2-D convolutional neural network framework for hyperspectral image change detection. A mixed-affinity matrix is firstly designed, and subsequently a 2-D convolutional neural network is designed to learn the discriminant features effectively from the multisource data at a higher level so as to enhance the generalization ability of the proposed change detection algorithm.

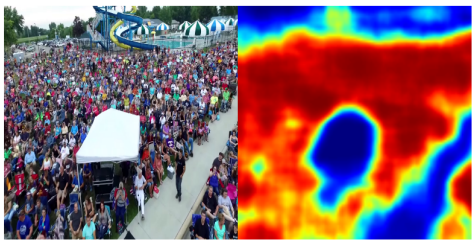


Fig. 1: Crowded image and the corresponding predicted heatmap of crowd presence.

Furthermore, since we deal with lightweight models that usually have inferior performance compared to the more complex ones, we focus on enhancing their performance. That is, the second goal of this work is to propose a novel regularization method in order to prevent over-fitting and enhance the generalization ability of the proposed lightweight models. Generally, this constitutes a major issue in deep learning algorithms, since neural networks are prone to over-fitting due to their high capacity. During the past years, several regularization schemes have been proposed in order to prevent overfitting in neural networks, ranging from common regularization methods, like $\mathscr{L}1/\mathscr{L}2$ regularization which penalize large weights during the network optimization, to Dropout [25] where for each training sample, a randomly selected subset of the activations is zeroed in each epoch, and a generalization of it, Dropconnect [26] which instead of activations, sets a randomly selected subset of weights within the network to zero. Other earlier works include weight elimination, [27], and Bayesian methods, [28]. From a different viewpoint, multitask-learning [29] constitutes a way of improving the generalization ability of a model. For example, in [30] the authors introduced techniques developed in semi-supervised learning in the deep learning domain. That is, they combined an unsupervised regularizer with a supervised learner to perform semi-supervised learning. Furthermore, in [31], a novel CNN architecture with an SVM classifier at every hidden layer is proposed. This companion objective acts as a kind of feature regularization. Finally, in [32] the authors propose a two-stage training method including pre-training process and implicit regularization training process, in order to address the overfitting problem. In the first stage a network model is trained to extract the image representation for anomaly detection, while in the second stage, the network is retrained, based on the anomaly detection results, to regularize the feature boundary and make it converge in the proper position. In this work, considering two-class classification problems, where the one class describes a specific concept, while the other class describes anything than this concept (i.e. genuine versus impostor class), *e.g.* Crowd and Non-Crowd, Face and Non-Face, etc., we propose the so-called *Class-Specific Discriminant (CSD)* regularizer, which exploiting the nature of the problem, aims to enhance the discriminative ability of the models, by enforcing data belonging to the class under consideration to be close to their class centroid. Finally, we should note that the proposed regularizer is applicable to several network architectures for binary classification problems, however its necessity is traced in improving the performance of real-time lightweight CNN models.

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

- We propose lightweight deep CNN models for various classification problems, capable of running in real-time on-drone.
- We propose a novel Class-Specific Discriminant regularizer in order to enhance the generalization ability of the proposed models.

The remainder of the manuscript is structured as follows. The proposed real-time CNN model architectures are provided in Section 2. Subsequently, the proposed CSD regularizer is presented in Section 3. The experiments, including the datasets description, the implementation details and the experimental results, are provided in Section 4. Finally, the conclusions are drawn in Section 5.

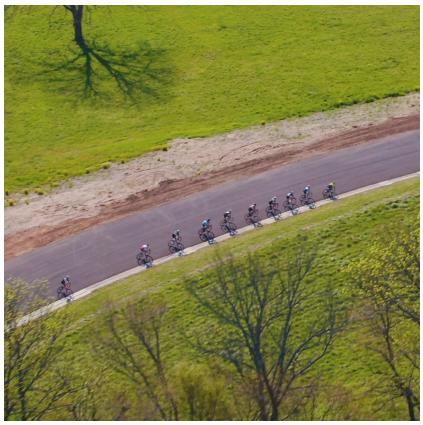
Table 1: VGG-720p - Input 32×32 / Input 64×64

Layer	Kernel	Stride	Pad	Max Pooling	Channels
conv1_1	3×3	1 / 1	1 / 1	- / -	16
conv1_2	3×3	1 / 1	1 / 1	✓/-	16
conv2_1	3×3	1 / 1	1 / 1	- / -	24
conv2_2	3×3	1 / 4	1 / 1	✓/ ✓	16
conv_last	8×8	1 / 1	0 / 0	- / -	2

2 Real-time CNN models

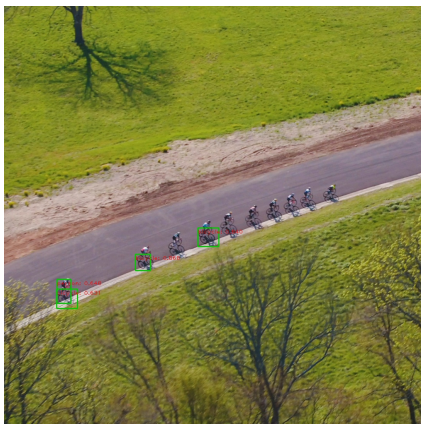
In this paper, we aim to propose effective deep models for various binary classification problems, which allow real-time deployment (about 25 frames per second) on-drone for high resolution images. We should highlight that it is crucial for the application to handle high resolution images, since objects in images captured by drones are extremely small, and thus image resizing in order to meet real-time deployment limits, would further shrink the object of interest, rendering the detection infeasible. Fig. 2 underlines the demand for high resolution images. That is, we provide an aerial image that contains bicycles (bicycles with bicyclists), Fig. 2a, and the resulting heatmaps for input of various resolutions, i.e. 640×480, 1280×720, and 1920×1080, utilizing a proposed model, Figs. 2d-2f. As we can observe as the resolution increases, we can achieve better performance. Furthermore, in Figs. 2b and 2c, we provide the predictions for the same input utilizing two state-of-the-art detectors which have trained to detect among other classes, also persons and bicycles, that is YOLO v.2 [15] and Faster R-CNN [33], which operate for input 604×604 and 1000×600 respectively. We should note that we provide the comparisons with the aforementioned state-of-the art detectors which provide bounding boxes as output, as opposed to the provided resulting heatmaps imposed by the application, in order to evaluate the accuracy of the proposed models against state-of-the-art models. As we can see, both the state-of-the-art detectors perform poorly, while they also run at much less than real-time, as we mention below. Note also, that SSD [16] and SSD with MobileNets [8], which operate for input 300×300 as well as for input 512×512, fail to detect any bicycle.

The objective of this work is two-fold: a) to propose real-time architectures that can be deployed on-drone, and b) to improve the state-of-the-art performance using CSD regularization. Thus, we propose two models consisting of only five convolutional layers, by discarding the deepest layers and pruning filters of the widely used VGG-16 model [34]. That is, we use the first four convolutional layers of the VGG-16 model with pruned filters, while the last convolutional layer consists of two channels, each for a class, since we deal with binary classification problems. The first model runs in real-time on-drone for 720p (1280×720) resolution image and the second one runs in real-time for 1080p (1920×1080) resolution image. Thus, we abbreviate the models, based on this attribute, as VGG-720p and VGG-1080p, respectively. Details on the proposed model architectures for both 32×32 and 64×64 input dimensions can be found in Table 1 and Table 2, for the VGG-720p and VGG-1080p models respectively. The models for the two cases use same kernels and channels, and use appropriate stride and pooling to achieve real-time deployment, as it is shown in the Tables. The evaluation results on the deployment speed for the proposed models are provided in the Experiments Section.

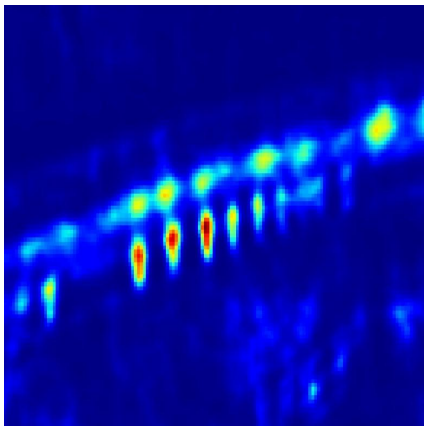


(a) Test image

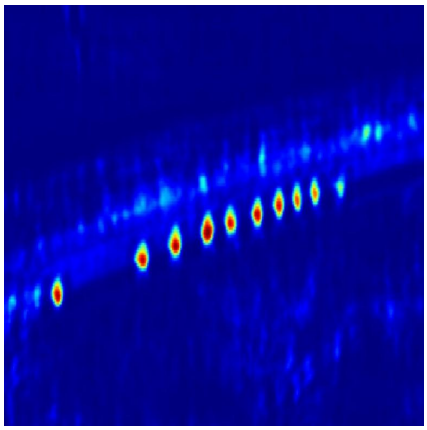
(b) YOLO v.2 Prediction for input of size 604×604



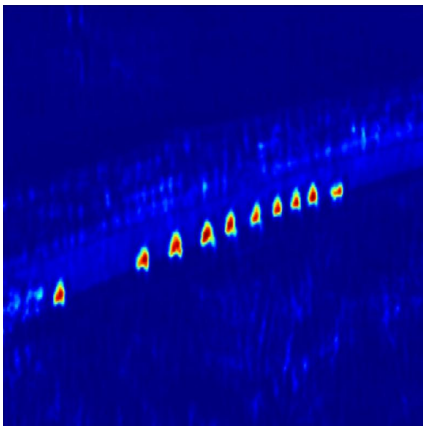
(c) Faster R-CNN Prediction for input of size 1000×600



(d) Resulting heatmap for input of size 640×480, utilizing the proposed VGG-1080p model



(e) Resulting heatmap for input of size 1280×720, utilizing the proposed VGG-1080p model



(f) Resulting heatmap for input of size 1920×1080, utilizing the proposed VGG-1080p model

Fig. 2: An aerial high resolution image containing bicycles (2a), predictions utilizing the YOLO v2 and the Faster R-CNN detectors (2b)-(2c), and the resulting heatmaps for various deployment resolutions utilizing the proposed VGG-1080p model trained for bicycle detection (2d)-(2f).

Table 2: VGG-1080p - Input 32×32 / Input 64×64

Layer	Kernel	Stride	Pad	Max Pooling	Channels
conv1_1	3×3	2 / 1	0 / 0	- / -	8
conv1_2	3×3	1 / 2	0 / 0	✓/ -	8
conv2_1	3×3	1 / 1	0 / 0	- / -	6
conv2_2	3×3	1 / 2	0 / 0	- / -	6
conv_last	8×8	1 / 1	0 / 0	- / -	2

3 Class-Specific Discriminant Regularizer

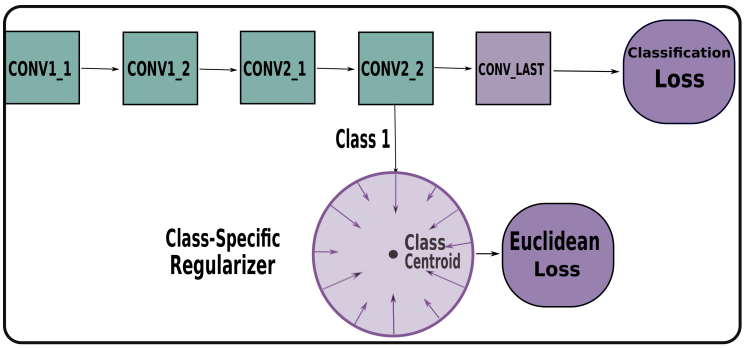


Fig. 3: Class-Specific Discriminant Regularizer

In this work, we propose a novel regularization method aiming to enhance the performance of the proposed lightweight real-time CNN models. The proposed regularization method aims to exploit the nature of the considered problems, that is, we investigate two-class problems, where the one class describes a specific concept, while the other class describes anything than this concept (i.e. genuine versus impostor class). The proposed regularizer traces its origins to Linear Discriminant Analysis (LDA) [35] based methods, [12], however based on the extremely wide variation of the impostor class, we exploit class-specific concepts, [36]. Towards this end, while the classifier aims to distinguish samples belonging to different classes, we propose to enhance the genuine class discrimination, by demanding the representations of the feature space generated by a specific deep neural layer belonging to the genuine class, to come closer to the class centroid. The L_2 norm can be used as similarity measure. The additional CSD criterion acts as regularizer to the classification objective. We could also demand for the remaining class to be away from the genuine class centroid, however we do not proceed in this direction, since the between class separability is preserved by the classification objective.

Thus, for an input space $\mathcal{X} \subseteq \Re^d$ and an output space $\mathcal{F} \subseteq \Re^q$, we denote as $\phi(\cdot;\mathcal{W}) : \mathcal{X} \to \mathcal{F}$ a deep neural network with $N_L \in \mathbb{N}$ layers, and set of weights $\mathcal{W} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{N_L}\}$, where $\boldsymbol{W}_l$ are the weights of a specific layer l. We also denote the set of weights up to layer l as $\mathcal{W}^l = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_l\}$. Then, the output of layer l for a given input $\boldsymbol{x}_i$ is computed as follows: $\phi(\boldsymbol{x}_i;\mathcal{W}^l) = \sigma_l(\boldsymbol{W}_l \cdot \phi(\boldsymbol{x}_i;\mathcal{W}^{l-1}) + \boldsymbol{b}_l)$, where

$\sigma_l(\cdot)$ is the activation function of layer l, $\boldsymbol{b}_l$ the bias term, $\phi(\boldsymbol{x}_i\,;\mathcal{W}^{l-1})$ the output of the previous layer, and $\cdot$ denotes a linear operation (*e.g.* matrix multiplication or convolution). Hence, we consider a set $\mathcal{D}_N = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of training samples on $\mathcal{X}$, and their corresponding representations, $\phi(\boldsymbol{x}_i\,;\mathcal{W}^l)$, at the layer l. Each sample is associated with a class label $c_i = \{0, 1\}$, where the class 1 corresponds to the genuine class. We also consider as $\mathcal{S} = \{(\boldsymbol{x}_i, c_i) : c_i = 1\}$ the set of samples belonging to the genuine class. Then we define the following objective:

$$\min_{\mathcal{W}^l} \mathcal{J}_{CSD} = \min_{\mathcal{W}^l} \sum_{\boldsymbol{x}_i \in \mathcal{S}} \|\phi(\boldsymbol{x}_i\,;\mathcal{W}^l) - \boldsymbol{\mu}_c\|_2^2, \tag{1}$$

where $\boldsymbol{\mu}_c = \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x}_j \in \mathcal{S}} \phi(\boldsymbol{x}_j\,;\mathcal{W}^l)$. Optimizing objective (1) lets the network learn parameters such that data samples of the genuine class are closely mapped to their class centroid. The Euclidean loss layer (Sum of Squares) is used to implement the regularizer. The proposed regularizer can be attached to one or multiple neural layers. Thus, for a deep neural model of N_L layers, the total loss in the regularized training scheme is computed by summing the above losses:

$$L_{total} = L_{classification} + \sum_{i=1}^{N_L} \eta_i L_e, \tag{2}$$

where the parameter $\eta_i \in [0, 1]$ controls the relative importance of the Euclidean loss of each deep layer. In our experiments we attach it to the so-called $CONV2\_2$ layer, as depicted in Fig. 3. Either Hinge loss or Cross Entropy loss can be utilized as classifiers. In our experiments we use the Hinge loss. We solve the above optimization problem using gradient descent. We finally note that it is straightforward to show that the optimization problem in eq. (1) can be reformulated as an accumulation of minimization of pairwise distances, [37], that is,

$$\min_{\mathcal{W}^l} \mathcal{J}_{CSD} = \min_{\mathcal{W}^l} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{S}} \|\phi(\boldsymbol{x}_i\,;\mathcal{W}^l) - \phi(\boldsymbol{x}_j\,;\mathcal{W}^l)\|_2^2, \tag{3}$$

and thus the proposed regularizer can also be implemented in terms of mini-batch training.

4 Experiments

In this Section, we present the experiments conducted in order to evaluate the proposed models regarding the deployment speed as well as the proposed regularization method. Throughout this work, we evaluate the detection speed in terms of frames per second (FPS), while we use Test Accuracy (Classification Accuracy) to evaluate the proposed regularizer, since we deal with balanced datasets. Each experiment is repeated five times, and we report the mean value and the standard deviation, considering the maximum value of the Test Accuracy for each experiment. The probabilistic factor is the random weight initialization. We also provide the curves of the mean Test Accuracy. Finally, comparisons are conducted against common regularization approaches, that is $\mathscr{L}1$, $\mathscr{L}2$, and Dropout.

4.1 Datasets

In order to evaluate the performance of the proposed CSD regularizer we conduct experiments on four datasets, constructed for Crowd, Football Player, Face, and Bicycle detection. The datasets' descriptions follow below.

4.1.1 Face

The dataset contains 70,000 train images of faces and equal number of train images of non-faces, and correspondingly a test set of 7,468 equally distributed face and non-face images. Images of faces have been randomly selected from the AFLW [38], MTFL [39], and WIDER FACE [40] datasets. Input images are of size 32×32. Sample images of the constructed *Face* dataset are presented in Fig. 4a.

4.1.2 Football Player

The dataset consists of 98,000 train images that contain equal number of football players and non-football players, and a test set of 10,000 images that also contain equal number of football players and non-football players. Input images are of size 32×32. Sample images of the *Football Player* dataset are illustrated in Fig. 4b.

4.1.3 Crowd-Drone

The dataset contains 40,000 drone-captured train images of equal number of crowded scenes and non-crowded scenes, and 11,550 equally distributed crowded and non-crowded test images. Input images are of size 64×64. Sample images of the constructed *Crowd-Drone* dataset are presented in Fig. 4c.

4.1.4 Bicycles

The dataset contains 51,200 equally distributed train images of bicycles (bicycle with bicyclist) and non-bicycles, and correspondingly a test set of 10,000 images. Input images are of size 64×64. Sample images of the constructed *Bicycles* dataset are presented in Fig. 4d.

4.2 Implementation Details

All the experiments conducted using the Caffe Deep Learning framework [41]. We use the mini-batch gradient descent for the networks' training. The learning rate (lr) is set to 10^{-4}, except for the Football dataset on the VGG-720p case, where it is set to 10^{-5} (since setting lr to 10^{-4} lead to unstable performance) and the batch size is set to 256. The momentum is 0.9. All the models are trained on an NVIDIA GeForce GTX 1080 with 8GB of GPU memory for 100 epochs, and can run in real-time when deployed on an NVIDIA Jetson TX2. The parameter η in eq. (2) for controlling the relative importance of the regularization loss is set to 0.001. Best results are printed in bold.

(a) Face Dataset. (b) Football Player Dataset.

(c) Crowd-Drone Dataset. (d) Bicycles Dataset.

Fig. 4: Sample images of the utilized datasets.

4.3 Experimental Results

First, we provide the evaluation results of the proposed models regarding the deployment speed. We test 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. Additionally, in order to accelerate the deployment speed and achieve real-time deployment, we utilize TensorRT[1], deep learning inference optimizer. TensorRT is a library that optimizes deep learning models providing FP32 (default) and FP16 optimizations for production deployments of various applications. In Table 3 we provide the detection speed in terms of FPS for the two proposed architectures and their corresponding image resolution on the NVIDIA Jetson TX2 module without the utilization of the TensorRT optimizer, with the TensorRT on the default mode, and finally with TensorRT on the FP16 mode. As we can see TensorRT and in particular the FP16 mode significantly accelerates the proposed models, achieving detection in-real time for high-resolution images. To gain some intuition about the deployment speed, we note that state-of-the-art detectors run at notably fewer FPS on Jetson TX2, and also for lower resolution input images. For example, SSD [16] runs at 6 FPS, for input of size 300×300, SSD with MobileNets [8] runs at 0.66 FPS for the same input, and YOLO v.2 [15] runs at 10 FPS for input of size 308×308, while it runs at 3.1 FPS for input of size 604×604. Finally, the Faster R-CNN [33] runs at 0.9 fps on the Jetson TX2 module. Finally, we should highlight that the deployment speed regards all the models, that is with and without the proposed regularizer, since the regularizer does not affect the deployment speed. That is, based on the Tables 6 and 7 which provide the accuracy rates of the aforementioned real-time models, the proposed VGG-720p model for crowd detection, where the input images are of size 64×64 runs at 25 fps with the utilization of TensorRT-FP16, and so does the corresponding baseline model, trained only with hinge loss, for the task of crowd detection. Correspondingly, the pro-

[1] `https://developer.nvidia.com/tensorrt`

Table 3: Speed (FPS)

Input	Model	Jetson TX2	TensorRT-FP32	TensorRT-FP16
32×32	VGG-720p	10.1	18.1	26.3
32×32	VGG-1080p	12.3	16.9	25.7
64×64	VGG-720p	8.7	16.6	25
64×64	VGG-1080p	8.8	18.5	25.6

posed VGG-1080p model for face detection, where the input images are of size 32×32 runs at 25.7 fps with the utilization of TensorRT-FP16, and so does its corresponding baseline model. The proposed models will be publicly available at the final version of the paper.

As we have already mentioned the parameter η in eq. (2) for controlling the relative importance of the regularization loss is set to 0.001, since in general performs better. A common choice for the $\mathscr{L}1$ and $\mathscr{L}2$ regularizers is the value 0.0005. However, we also conduct more extensive experiments for the evaluation of the CSD regularizer against the common $\mathscr{L}1$ and $\mathscr{L}2$ at various meaningful regularization factors. In Table 4 we provide the evaluation results for the Face dataset utilizing VGG-720p model. As we can see, the proposed regularizer accomplishes consistently superior performance against the compared regularizers. It can also be noticed that as the regularization factor increases in the $\mathscr{L}1$ regularizer, the performance considerably drops, while for bigger values, the model cannot even be trained. In the following, we set the regularization factor for each of the regularizers to their optimal values, that is the regularization factor of $\mathscr{L}1$ is set to 0.0001, of $\mathscr{L}2$ to 0.0005, and of CSD to 0.001.

Table 4: Face Dataset - VGG-720p model: Comparisons against $\mathscr{L}1$ and $\mathscr{L}2$ regularizers for different values of regularization factor.

Regularizer	0.0001	0.0005	0.001	0.005	0.01	0.05	0.1
$\mathscr{L}1$	0.9186 ± 0.0017	0.8984 ± 0.0092	0.7557 ± 0.01871	0.5489 ± 0.059	0.56 ± 0.0049	0.5063 ± 0.00141	0.5056 ± 0.005
$\mathscr{L}2$	0.9180 ± 0.0032	0.9204 ± 0.0021	0.9189 ± 0.0056	0.9178 ± 0.0016	0.9162 ± 0.0012	0.8988 ± 0.0013	0.8788 ± 0.002
CSD	**0.9224 ± 0.0044**	**0.9227 ± 0.0049**	**0.9253 ± 0.0028**	**0.9227 ± 0.0021**	**0.9237 ± 0.0015**	**0.9240 ± 0.004**	**0.9242 ± 0.0013**

Subsequently, as we have already mentioned, either Hinge loss or Cross Entropy loss, can be utilized as classification losses. In our experiments we used Hinge loss, since we have seen that performs better, however, we also provide evaluation results utilizing the Cross Entropy, while we also provide indicative comparisons with the common $\mathscr{L}1$ and $\mathscr{L}2$ regularizers, as well as for the Dropout regularization (with the default probability value: 0.5) for the Face dataset, utilizing the VGG-720p model. As we observe in Table 5, the proposed CSD regularizer improves the baseline performance for both the considered classification losses. We also see that Dropout achieves improved performance, $\mathscr{L}2$ regularizer marginally improves the performance, while the $\mathscr{L}1$ regularizer harms the performance. We can also observe that the proposed CSD regularizer is superior over the common $\mathscr{L}1$ and $\mathscr{L}2$ regularizers, as well as over Dropout. Furthermore, since the proposed CSD regularizer can be combined with the aforementioned regularizers, we also perform experiments utilizing the $\mathscr{L}2$ and Dropout regularizers (which improve the baseline performance) in combination with

the CSD regularizer, and we observe that we can further improve the performance over the baseline, as well as over each individual regularization method.

Table 5: Face Dataset - VGG-720p model: Test Accuracy

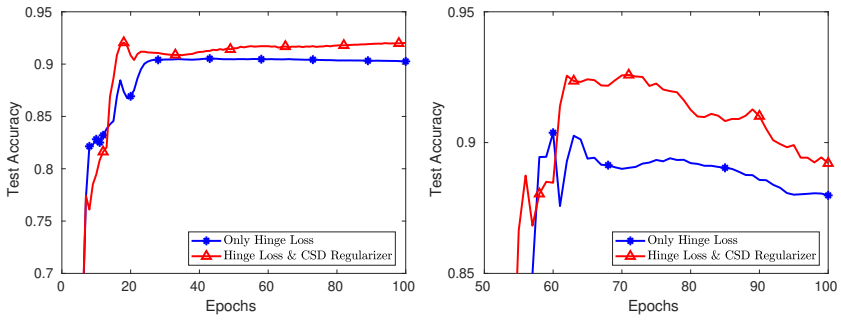
Training Approach	**Test Accuracy**
Only Hinge Loss	0.9191 ± 0.0037
Hinge Loss & $\mathscr{L}1$ Regularizer	0.9186 ± 0.0017
Hinge Loss & $\mathscr{L}2$ Regularizer	0.9204 ± 0.0021
Hinge Loss & Dropout	0.9224 ± 0.002
Hinge Loss & CSD Regularizer	**0.9253 ± 0.0028**
Hinge Loss & $\mathscr{L}2$ & CSD Regularizer	0.9266 ± 0.002
Hinge Loss & Dropout & CSD Regularizer	**0.9268 ± 0.0015**
Only Cross Entropy Loss	0.8984 ± 0.0092
Cross Entropy Loss & CSD Regularizer	**0.9098 ± 0.0033**

In Table 6, we present the mean value and the standard deviation of the Test Accuracy, for the considered training approaches, that is utilizing only Hinge loss, and Hinge loss with the proposed CSD regularizer, as well as with the compared regularizers (that is $\mathscr{L}1$, $\mathscr{L}2$, and Dropout) on all the utilized datasets for the VGG-720p case, while in Table 7 we provide the corresponding evaluation results for the VGG-1080p case. From the demonstrated results several remarks can be drawn. First, we can observe that the proposed regularizer considerably improves the baseline performance utilizing both the VGG-720p and VGG-1070p in all the utilized datasets. Furthermore, we can see that the CSD regularizers is superior against all the compared regularizers, in all the considered cases, except for one case, that is the Football Player dataset for the VGG-720p model, where the Dropout regularizer marginally outperforms the proposed regularizer. In addition, we can see that the Dropout regularizer improves the performance of the baseline model in the most considered cases while it harms the baseline performance in two out of eight cases. $\mathscr{L}2$ regularizer improves the performance only in the half of the considered cases, while the $\mathscr{L}1$ one generally harms the performance, except for two cases. Correspondingly, in Figs. 5-8 we provide the comparison of the mean Test Accuracy of the only Hinge loss training against Hinge loss & CSD regularized training on all the utilized datasets for both the proposed model architectures, where the improved performance is also depicted.

Table 6: Test Accuracy: VGG-720p

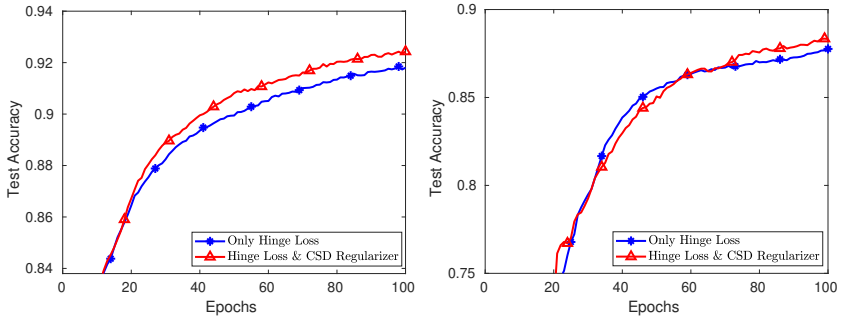
Training Approach	**Crowd-Drone**	**Football Player**	**Face**	**Bicycles**
Only Hinge Loss	0.9327 ± 0.0089	0.9734 ± 0.007	0.9191 ± 0.0037	0.9684 ± 0.0029
Only Hinge Loss & $\mathscr{L}1$	0.9316 ± 0.009	0.9734 ± 0.0088	0.9186 ± 0.0017	0.9638 ± 0.0039
Only Hinge Loss & $\mathscr{L}2$	0.9238 ± 0.0066	0.9814 ± 0.0013	0.9204 ± 0.0021	0.9668 ± 0.0025
Only Hinge Loss & Dropout	0.9227 ± 0.0021	**0.9830 ± 0.0023**	0.9224 ± 0.002	0.9716 ± 0.0031
Hinge Loss & CSD regularizer	**0.9399 ± 0.0087**	0.9820 ± 0.0024	**0.9253 ± 0.0028**	**0.9722 ± 0.0026**

Furthermore, in order to validate our claim that the proposed regularizer is applicable to various network acrhitectures for binary classification problems, we also perform experiments utilizing a much more complex model than the proposed real-time



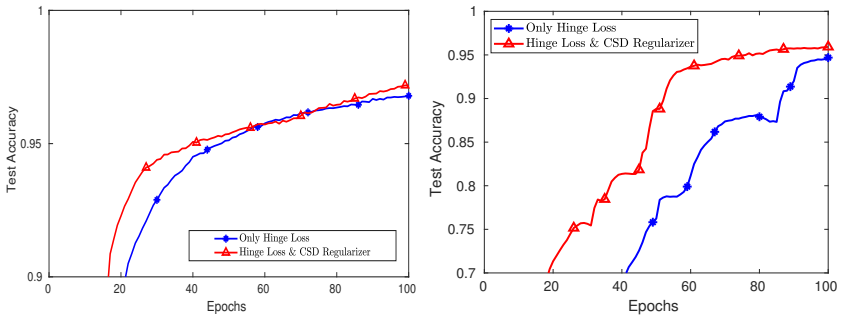
(a) VGG-720p (b) VGG-1080p

Fig. 5: Crowd-Drone



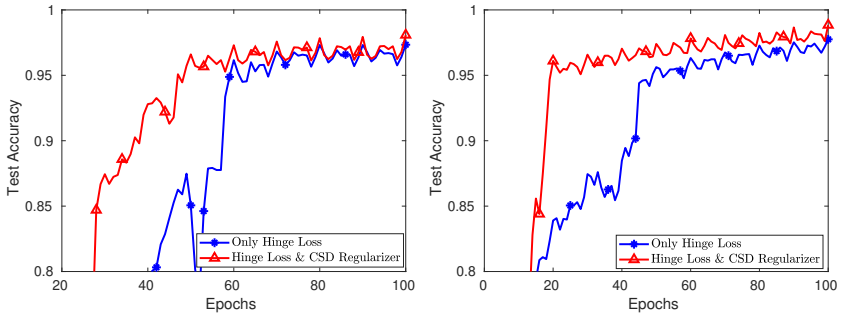
(a) VGG-720p (b) VGG-1080p

Fig. 6: Face



(a) VGG-720p (b) VGG-1080p

Fig. 7: Bicycles



(a) VGG-720p (b) VGG-1080p

Fig. 8: Football Player

Table 7: Test Accuracy: VGG-1080p

Training Approach	Crowd-Drone	Football Player	Face	Bicycles
Only Hinge Loss	0.9270 ± 0.027	0.9785 ± 0.007	0.8787 ± 0.0015	0.9479 ± 0.0153
Only Hinge Loss & $\mathscr{L}1$	0.9220 ± 0.012	0.8757 ± 0.0176	0.8814 ± 0.0057	0.8699 ± 0.01339
Only Hinge Loss & $\mathscr{L}2$	0.9296 ± 0.0095	0.9730 ± 0.0095	0.8801 ± 0.009	0.9460 ± 0.009
Only Hinge Loss & Dropout	0.9284 ± 0.0199	0.9804 ± 0.005	0.8682 ± 0.0029	0.9535 ± 0.077
Hinge Loss & CSD regularizer	**0.9380 ± 0.0026**	**0.9884 ± 0.004**	**0.8828 ± 0.0024**	**0.9603 ± 0.0025**

models. That is, we utilize the common VGG-16 [34] model, discarding only the fully connected layers and we apply the proposed regularizer. We should highlight that the utilized modified fully convolutional VGG-16 model (abbreviated as FC-VGG16) is out of memory even in a GTX 1080 for high-resolution input. Furthermore we perform comparisons against competitive regularizers (i.e. $\mathscr{L}1$, $\mathscr{L}2$, and Dropout). In Table 8 we present the evaluation results utilizing the FC-VGG16 model in the Face dataset. As we can observe the CSD regularizer improves the baseline performance, not only utilizing lightweight models as the specific application imposes, but also utilizing more complex models. Furthermore, we can observe that the CSD regularizer is superior over all the compared ones.

Table 8: Face Dataset: FC-VGG16

Training Approach	Test Accuracy
Only Hinge Loss	0.9393 ± 0.0215
Only Hinge Loss & $\mathscr{L}1$	0.9348 ± 0.0001
Only Hinge Loss & $\mathscr{L}2$	0.9463 ± 0.0033
Only Hinge Loss & Dropout	0.9469 ± 0.0054
Hinge Loss & CSD Regularizer	**0.9498 ± 0.004**

Finally, we conducted a post-hoc Bonferroni test [42], for ranking the proposed regularization method and the compared regularizers and evaluating the statistical significance of the obtained results. The performance of two methods is significantly different, if the corresponding average ranks over the datasets differ by at least the critical difference (CD):

$$CD = q_a \sqrt{\frac{m(m+1)}{6D}}, \tag{4}$$

where m is the number of methods compared, D is the number of datasets and critical values q_α can be found in [42]. In our comparisons we set $\alpha = 0.05$. We consider as number of the datasets as eight in the performed test, since we evaluate the performance of all the training approaches using both the proposed models (i.e. VGG-720p and VGG-1080p). The compared methods are five, that is the proposed regularizer, as well as the compared regularizers (i.e. $\mathscr{L}1$, $\mathscr{L}2$, Dropout) which are compared with a control method which is the only hinge loss training approach. The ranking results are illustrated in Fig. 9. The vertical axis depicts the five methods, while the horizontal axis depicts the performance ranking. The circles indicate the mean rank and the intervals around them indicate the confidence interval as this is determined by the CD value. Overlapping intervals between two methods indicate that there is not a statistically significant difference between the corresponding ranks, while non-overlapping

intervals indicate that the compared methods are significantly different. As we can observe, the proposed regularizer is significantly different against the control method, that is the only hinge loss training approach, while none of the compared regularizers is statistically different against the baseline approach. We could also note the proposed regularizer is also statistically different against the $\mathscr{L}1$ and $\mathscr{L}2$ regularizers.

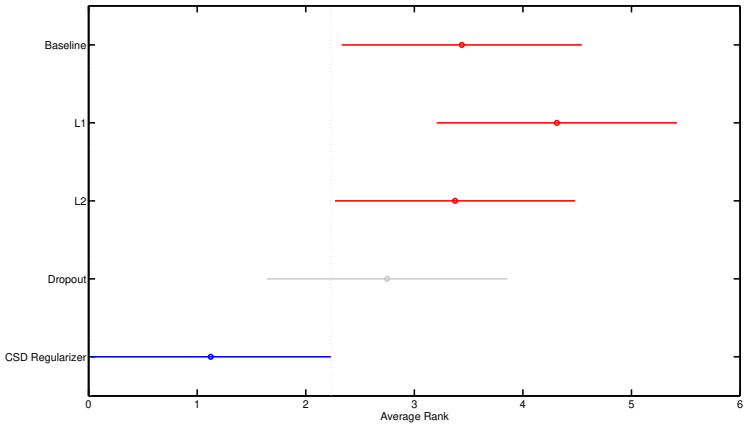


Fig. 9: Post-Hoc Bonferroni Test

5 Conclusion

In this paper, lightweight models capable of running on-drone for high-resolution video input, for various binary classification problems have been proposed, in the context of media coverage of certain sport events by drones. Subsequently, a novel Class-Specific Discriminant regularizer was proposed, in order to improve the generalization ability of the proposed real-time models, exploiting the nature of the considered two-class problems. The experimental evaluation on four datasets indicated the effectiveness of the proposed regularizer in enhancing the generalization ability of the proposed models.

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.

References

1. B Boser Le Cun, John S Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems 2*, pages 396–404. Morgan Kaufmann Publishers Inc., 1990.

2. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
3. Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.
4. Maria Tzelepi and Anastasios Tefas. Deep convolutional learning for content based image retrieval. *Neurocomputing*, 275:2467–2478, 2018.
5. Yunlong Yu, Fuxian Liu, and Sheng Mao. Fingerprint extraction and classification of wireless channels based on deep convolutional neural networks. *Neural Processing Letters*, pages 1–9, 2018.
6. Chen Liu, Weiyan Hou, and Deyin Liu. Foreign exchange rates forecasting with convolutional neural network. *Neural Processing Letters*, 46(3):1095–1119, 2017.
7. Yanyi Liu, Wenbo Liu, and Yin Wu. Associative memory realized by reconfigurable coupled three-cell cnns. *Neural Processing Letters*, 48(2):1123–1134, 2018.
8. Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
9. Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
10. A Claesson, D Fredman, L Svensson, M Ringh, J Hollenberg, P Nordberg, M Rosenqvist, T Djarv, S Osterberg, J Lennartsson, et al. Unmanned aerial vehicles (drones) in out-of-hospital-cardiac-arrest. *Scandinavian journal of trauma, resuscitation and emergency medicine*, 24(1):124, 2016.
11. Ludovic Apvrille, Tullio Tanzi, and Jean-Luc Dugelay. Autonomous drones for assisting rescue services within the context of natural disasters. In *General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI*, pages 1–4. IEEE, 2014.
12. Maria Tzelepi and Anastasios Tefas. Human crowd detection for drone flight safety using convolutional neural networks. In *European Signal Processing Conference (EUSIPCO), Kos, Greece, 2017*.
13. Nikolaos Passalis, Anastasios Tefas, and Ioannis Pitas. Efficient camera control using 2d visual information for unmanned aerial vehicle-based cinematography. In *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*, pages 1–5. IEEE, 2018.
14. Nikolaos Passalis and Anastasios Tefas. Deep reinforcement learning for controlling frontal person close-up shooting. *Neurocomputing*, 335:37–47, 2019.
15. Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.
16. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
17. Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.
18. Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.
19. Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2752–2761, 2018.
20. Dawei Li, Xiaolong Wang, and Deguang Kong. Deeprebirth: Accelerating deep neural network execution on mobile devices. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
21. Jiaqi Shao, Changwen Qu, Jianwei Li, and Shujuan Peng. A lightweight convolutional neural network based on visual attention for sar image target classification. *Sensors*, 18(9):3039, 2018.
22. Yi Yang, Hengliang Luo, Huarong Xu, and Fuchao Wu. Towards real-time traffic sign detection and classification. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):2022–2031, 2016.
23. Yuanjun Huang, Xianbin Cao, Qi Wang, Baochang Zhang, Xiantong Zhen, and Xuelong Li. Long-short term features for dynamic scene classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.
24. Qi Wang, Zhenghang Yuan, Qian Du, and Xuelong Li. Getnet: A general end-to-end 2-d cnn framework for hyperspectral image change detection. *IEEE Transactions on Geoscience and Remote Sensing*, (99):1–11, 2018.

25. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
26. Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.
27. Andreas S Weigend, David E Rumelhart, and Bernardo A Huberman. Generalization by weight-elimination with application to forecasting. In *Advances in neural information processing systems*, pages 875–882, 1991.
28. David JC MacKay. Probable networks and plausible predictionsâ"a review of practical bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995.
29. Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
30. Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th international conference on Machine learning*, pages 1168–1175. ACM, 2008.
31. Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015.
32. Qinghe Zheng, Mingqiang Yang, Jiajie Yang, Qingrui Zhang, and Xinxin Zhang. Improvement of generalization ability of deep cnn via implicit regularization in two-stage training process. *IEEE Access*, 6:15844–15869, 2018.
33. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
34. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
35. Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.
36. Georgios Goudelis, Stefanos Zafeiriou, Anastasios Tefas, and Ioannis Pitas. Class-specific kernel-discriminant analysis for face verification. *IEEE Transactions on Information Forensics and Security*, 2(3):570–587, 2007.
37. Marios Kyperountas, Anastasios Tefas, and Ioannis Pitas. Salient feature and reliable classifier selection for facial expression classification. *Pattern Recognition*, 43(3):972–986, 2010.
38. Martin Koestinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2144–2151. IEEE, 2011.
39. Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.
40. Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5525–5533, 2016.
41. Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.
42. Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.