CrowdCounting: MobileCount on VisDrone dataset

Pasquale De Marinis
dept. of Computer Science
Università degli Studi di Bari
Bari, Italy
p.demarinis6@studenti.uniba.it

Mauro Giuseppe Camporeale
dept. of Computer Science
Università degli Studi di Bari
Bari, Italy
m.camporeale18@studenti.uniba.it

***Abstract*—Contribution: This work focuses on finding a light-weight deep learning model able to solve the problem of Crowd counting while also having a good performance on a mobile device like a drone. Background: The necessity to provide the drone a density map of people underneath its route so that it can safely perform an emergency landing. Research Questions: The main aim of this study is to find a light-weight neural network model that gives as output both a density map and the count estimation of the people in the scene; the model has to be able to process a good amount of frames per second thus producing a real-time estimation of the input image. Methodology: The best light-weight models taken from the Awesome Crowd Counting repository were examined and tested, then the model that was considered the best was fitted to the VisDrone dataset. Findings: Mobile Count fully convolutional neural network has proved itself has the best light-weight model, as it has a very good accuracy in crowd counting while retaining nice performances in terms of Frames Per Second.**

***Index Terms*—computer vision, drone, crowd counting, light weight.**

I. Introduction

Crowd counting, as the name suggests, is the task to estimate the number of people in congested scenes. It has gained a lot of interests in recent years due to its significant importance in surveillance and security applications. As in many computer vision tasks, state-of-the-art approaches for crowd counting are also based on deep neural networks. The approaches to tackle this problem could be classified into three categories: detection-based [1], regression-based [2] and density map estimation-based methods [3], [4], [5], [6], [7], [8], [9].

The crowd counting problem is closely related to mobile or embedded systems, since these are the systems that most of the times are employed to face these kind of problems. In real applications with mobile or embedded systems, it is almost equivalently important to obtain an acceptable accuracy and to achieve fast inference speed with a limited computation budget. A variety of light-weight neural networks have been proposed for image classification [10], [11], [12], segmentation [13], [14], [15] and object detection [16], [17], [18]; however, light-weight crowd counting models like the ones in [19], [20] and [21] have been rarely studied, despite their importance. In our case, we focused on searching a light-weight model being able to process a good amount of frames per second (FPS), and so providing, with good accuracy, a real-time density map of the area under the drone thus providing the drone the ability to autonomously find a safe area in case of an emergency landing.

II. Related Works

The earliest approaches to the crowd counting problem were based on people detection [1], the entire person or only a part of it (head) had to be detected, so that number of detected objects can provide an estimate of the count of the crowd. Unfortunately, these methods often suffered from the person size being too small or the crowd being extremely dense and thus occluding people in it.

Regression-based method [2] were then developed; these methods directly computed the number of objects based on features extracted from cropped image patches. Despite the effectiveness in counting, these methods could not produce any other useful information on the input image.

Instead of making a regression on the overall of the entire image, density map estimation-based methods predict the density at each pixel and obtain the crowd count by summing over all pixels, these methods [3], [4], [5], [6], [7], [8], [9] can offer stronger supervision and preserve more fine-grained information. Zhang et al. [5] proposed a Multi-column Convolutional Neural Network (MCNN) based on density map estimation. The input of MCNN is an image and the output is a crowd density map whose integral gives the overall crowd count. Li et al; [8] proposed CSRNet, which is a single column Fully Convolutional Network (FCN) architecture with cascaded dilated convolutional layers; Sam et al. [7] trained a recursively growing CNN tree for routing input image patches to appropriate expert regressors; Shi et al. [22] proposed a pool of decorrelated regressors based on deep negative correlation learning and so on.

We were mostly interested in the most recent density map estimation-based methods. We focused on the ones listed below:

- ACSCP [21]: is a double Generative Adversarial Networks working both on the whole frame and on the same divided into 4 quadrants, when tried on the dataset cited in [21], expected performances could not be replicated;
- PeopleFlow [20]: a Convolutional Neural Network that counts people by comparing their movement between two successive frames; the pre-trained model was provided, but it performed much worse than as described in [20],

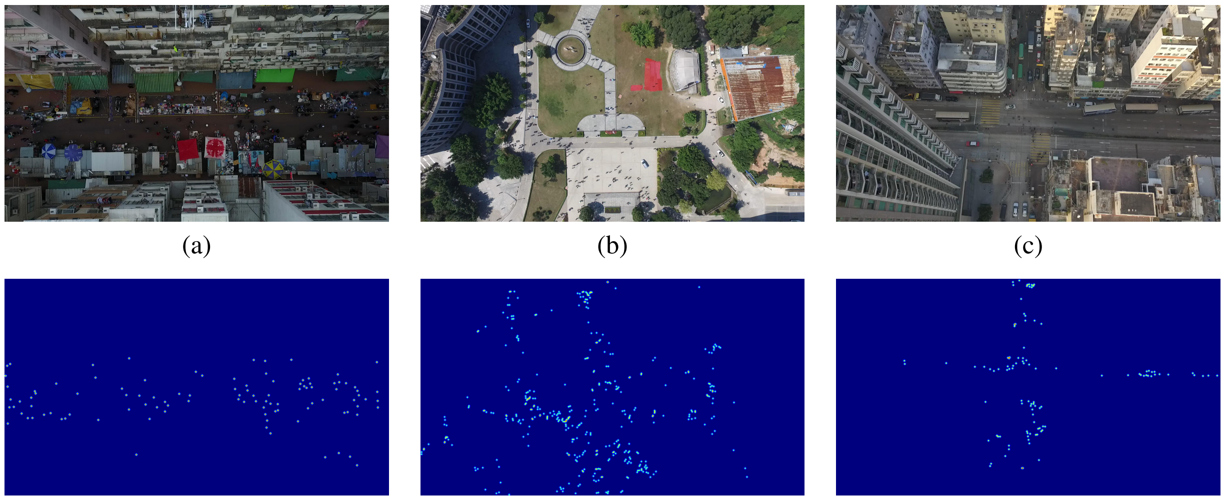

Fig. 1: Three frames of the Visdrone dataset (a, b, c) with the respectively generated heatmaps (d, e, f)

we were not able to train it from scratch as it requires an amount of VRAM that our setup could not handle;

- MobileCount [23]: a net that uses a light weight encoder combined with a light weight decoder, the reliability of the model was tested and it respects the performances described in [23] both in terms of accuracy and FPS.

The model we choose to use to tackle this task was the latter since, as in the above description, it was the most promising one.

III. The VisDrone Dataset

VisDrone [24] is a challenge whose purpose is to encourage the development of automatic understanding of visual data collected from drones, bringing computer vision to drones more and more closely. The challenge involves 3 different tasks dealing respectively with Object detection, Multi-Object tracking and CrowdCounting. For each challenge a different dataset is provided. In our project we used the VisDrone-Counting2020 (DroneCrowd) dataset.

The DroneCrowd dataset consists of 112 sequences, 82 for training, 30 for testing. Each video sequence is made of 30 high resolution frame (1920x1080) [Fig. 1]. The annotations are the (x, y) coordinates of the people heads for each frame.

IV. Data preprocessing

For our dataset, we made the basic preprocessing steps made for images. Values of the pixels were scaled in [0, 1] range, and we normalized it subtracting mean and standard deviation of the dataset.

A. Ground Truth Generation

Since the model we chose to address this task on needs to be trained both on an input image and its density map [Fig. 1], the first operation that had to be done was the generation of a density map starting from the single input image and its annotations; this was achieved by applying a Gaussian blur kernel at each annotated head position:

$$D(x) = \sum_{i=1}^{M} \delta(x - x_i) * G_{\sigma_i}(x) \tag{1}$$

where $\delta(\cdot)$ represents a delta function and $x_i, i = 1, ..., M$ denote the positions of the i-th annotated head; σ_i denotes the variance of the Gaussian kernel applied at position i.

B. Data augmentation

For augmenting data we use two techniques:

- Random horizontally flip
- Random gamma correction

The random horizontally flip, simply mirrors the image horizontally (so along the vertical axis). This technique allow us to double the images with negligible computational overhead. In our experiments we chose to flip an image with a probability $p = 0.5$.

The random gamma correction (GC), applies a non linear transformation of the image in order to move the image values towards darker or brighter values. The transformation is based on the power-law, so when we darken the image, darker region will be more shifted than brighter regions, and the opposite happens when we brighten the image. The formula is described in (2).

$$G(l) = c * l^{\gamma} \tag{2}$$

where l is the pixel value, that is multiplied by a constant c and elevated to the power γ. When $\gamma > 1$ the results are darker, and when $\gamma < 1$ the results are brighter. We choose $c = 1$ and $\gamma = 1/\xi$, where ξ is sampled from a $Beta(\alpha; \beta)$ distribution, with $\alpha = 4.2$ and $\beta = 2.4$. We choose this distribution and these parameters in order to obtain in most cases slightly darker images, and with lower probability darker images. This augmentation has the objective to fill up the lack of night images in the training set.

Operator	**Output size**	**t**	**n**	**s**	**Output Channels (c)**				
					MC (x0.5)	MC (x0.75)	MC	MC (x1.25)	MC (x2)
Image	1				3	3	3	3	3
Conv2d	1/2	–	1	2	16	32	32	64	64
MaxPool		–	1	2					
Bottleneck	1/4	1	1	1	16	32	32	64	64
Bottleneck	1/8	6	2	2	32	48	64	96	128
Bottleneck	1/16	6	3	2	64	80	128	160	256
Bottleneck	1/32	6	4	2	128	160	256	320	512

TABLE I: Different settings of the MobileCount architecture, the first and the second one are added in our work, the others are presented in [23]. Conv2d represents a 3×3 convolution layer. MaxPool represents a 3×3 max pooling layer. Each row describes an operation with n repeats, stride s, expansion factor t and output channels c. The different settings are obtained by changing the output channels c for individual operations.

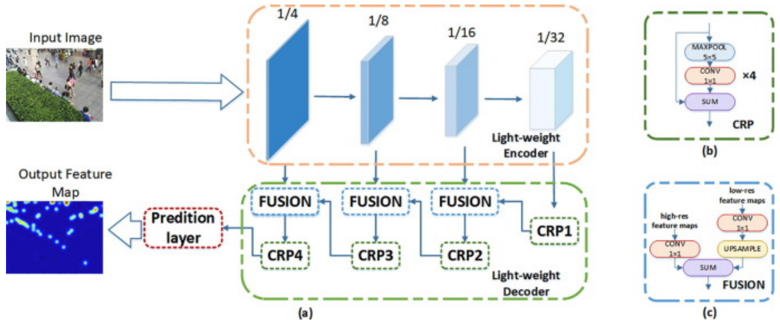

Fig. 2: The MobileCount overall architecture

V. Method

A. Network architecture

In this section we briefly explain the MobileCount [23] model architecture. MobileCount is a Fully Convolutional Network composed of two parts: an encoder and a decoder. The encoder is based on the MobileNetV2 [12], that is specifically designed for mobile applications, with significantly reduced memory footprint and improved classification accuracy. MobileNetV2 build a lightweight architecture based on a inverted residual structure having linear bottlenecks. A bottleneck is composed by the following operations:

- 1×1 Point-wise convolution
- 3×3 Depth-wise convolution
- 1×1 Point-wise convolution
- ReLU activation function

To reduce the FLOPs and the number of parameters without affecting the accuracy, MobileCount reduces the number of bottlenecks from 7 to 4, a number obtained by some empirical study. MobileNetV2 is so used as the backbone network that is tailored specifically for crowd counting. The model is even lighter thanks to a 3×3 MaxPooling of stride 2 that reduce the resolutions for the next bottlenecks layers. MobileCount offers 3 variants changing the number of feature channels, we added another two variants reducing even more the feature channels. The encoder architecture is shown in Fig. 3

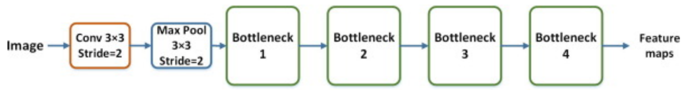

Fig. 3: The MobileCount encoder architecture

To correctly estimate the crowd density of a particular pixel, the information of the pixel itself is not enough, it is necessary to incorporate the context information of multiple scales to tackle different person sizes. MobileCount, chooses the Light-Weight RefineNet [15] decoder that was originally designed for semantic segmentation. This decoder has been specifically simplified from RefineNet [25] to achieve real-time performance, which can be combined with any encoder of multi-scale feature maps.

The decoding process of the Light-Weight RefineNet starts with the propagation of the last output from the backbone (with the lowest resolution) through chained residual pooling (CRP) block (Fig. 2 (b)) before being fed into a FUSION block (Fig. 2 (c)) along with the second to last feature map. Inside the FUSION block, each path is convolved with 1×1 convolution and the low-resolution feature maps are upsampled to the high-resolution among the paths. The two paths are then summed up, and in the same way, further fed into through several CRP and FUSION blocks until the desired resolution is reached. Subsequently, the generated feature maps are propagated into the prediction layer to produce the final density map.

The Prediction Layer is a regression-based layer which applies a 1×1 convolution, embed the d-dimensional feature vector at each pixel of an input feature map to a density value, and then the resulting density map is upsampled to the original image size by bilinear interpolation.

B. Loss functions

As a loss function, MobileCount uses the Mean Squared Error between the estimated density map matrix and the ground truth.

$$L_{den} = \frac{1}{N}\sum_{i=1}^{N}\left\|D^{PM}(i) - D^{GT}(i)\right\|_2^2 \qquad (3)$$

where N is the number of training samples; $D^{PM}(i)$ represents the predicted density map and $D^{GT}(i)$ represents the ground truth density map; i denotes the $i-th$ training sample; $\|\cdot\|_2$ represents the Euclidean distance.

VI. Experiments

In this section, we will firstly present the implementation details and introduce the adopted evaluation metrics. Secondly, we evaluate the performance of the net with different image sizes and optimizers. Finally, we compare our best model with state-of-the-art crowd counting approaches. Our models achieve comparable counting performance and real-time processing speed.

A. Implementation Details

The training and testing are performed on an NVIDIA GeForce GTX 1660 GPU and a Intel Core I7-9795H CPU @ 2.60Ghz using PyTorch framework. In our experiment, we used both Adam optimization and SGD optimization with a batch size of 2 for the standard architecture, 4 for the x0.75 and 4 for the x0.5 architecture. The weight decay rate is set to $1 \times 10e-4$. The initial learning rate is set to $1 \times 10e-4$.

The speed in term of FPS was also tested on the same setup.

B. Evaluation Metrics

Following [3], [8], [23], [26], we use Mean Absolute Error (MAE) and Mean Square Error (MSE) to evaluate our model. They are defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|C_i^{PM} - C_i^{GT}| \qquad (4)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(C_i^{PM} - C_i^{GT})^2} \qquad (5)$$

where N is the number of test images; C_i^{PM} is the predicted crowd count of each image obtained by summing over all pixels of the density map and C_i^{GT} is the real count.

C. Comparison of Different Setting

Various configurations were tested in order to better tune the model on the VisDrone dataset.

It is important to point out that the train-validation split has the proportion of 80-20 percent on the total number of frames in the train set; such technique was used since the first of our experiments, we also tried to split with the same proportions not on the frames but on the video sequences in the train set, so that the validation set contained completely new frames from the one in the train set, this gave a better estimation of the errors that the model would than obtain on the test set, but other than that nothing else changed.

First of all the main model was tested both with full size images (1920x1080) and with half sized images (960x540) [Table II].

Image Size	**MAE**	**RMSE**	**FPS**
(1920, 1080)	21.780	29.167	15.38
(960, 540)	36.413	51.141	34.44

TABLE II: Comparison of the results on MobileCount with different image size; FPS were calculated with a batch size = 4.

It can be seen that even thou the MAE and RMSE are higher on the halved-size image, the gain in term of FPS is great enough to compensate the accuracy lost, so we decided to carry out test on both the image size, as we came to the conclusion that both using the full-sized image and the half-sized image could be useful in different situations.

The next test was devote to investigate on what optimizer between ADAM and SGD could provide the best results in our case; we tested the net with full-size and half-sized image with both optimizers and compared the results [Table III].

Optimizers	**Image Size**	**MAE**	**RMSE**
ADAM	(1920, 1080)	21.780	29.167
ADAM	(960, 540)	36.413	51.141
SGD	(1920, 1080)	57.358	87.119
SGD	(960, 540)	30.287	38.428

TABLE III: Comparison of the results on MobileCount with different image sizes and different optimizers.

After this test we acknowledged that there was not a best optimizer, and that both had to be tested, although it seems that ADAM performs better on full sized images while SGD on the smaller ones.

Than we started testing the 3 different MobileNet architectures: MobileCount, MobileCountx1.25, MobileCountx2; these 3 nets differs one from each other only for the number of channels as showed in [Table I]. The test clearly shows how the lightest net was the one that performed the best both in term of accuracy and FPS, so we decided to create two more nets that we denominated MobileCountx0.75 and MobileCountx0.5 that had a number of channels even lower than the standard

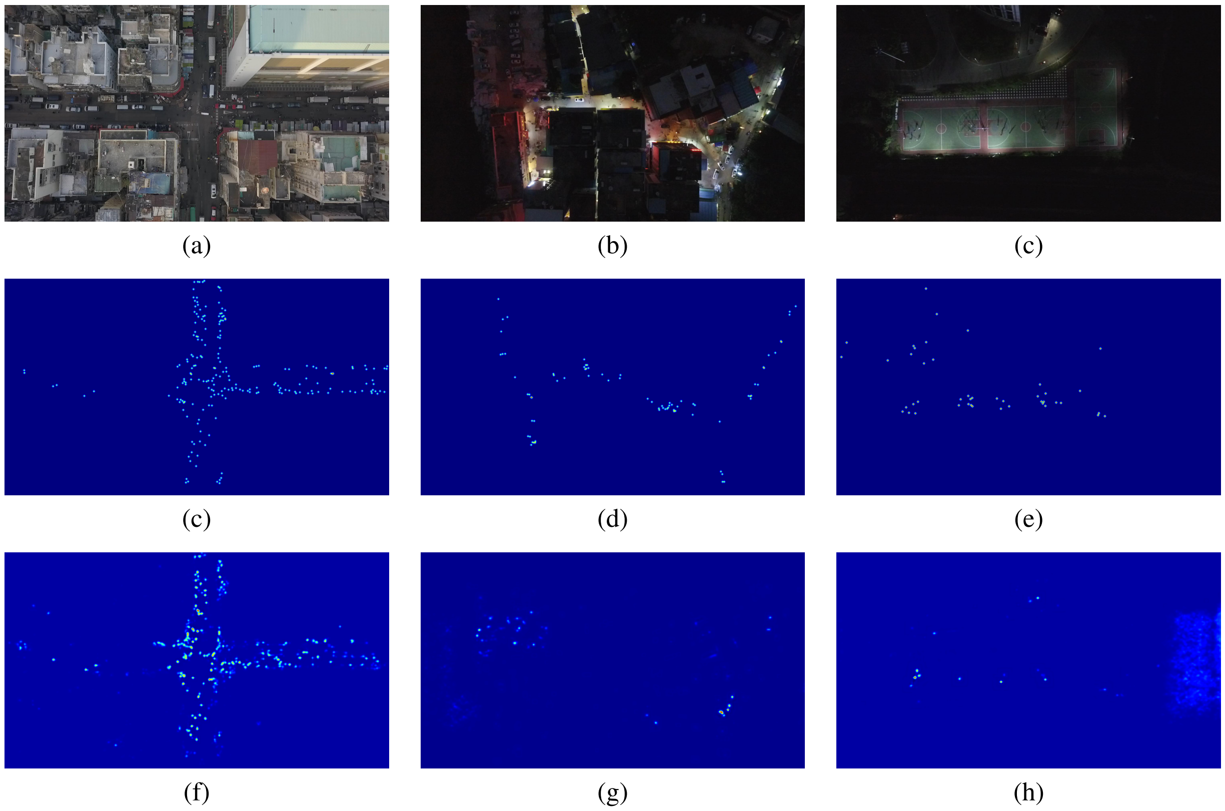

Fig. 4: Three frames of the Visdrone test set (a, b, c) with the respectively generated heatmaps (d, e, f) and the predictions of the best model (f, g, h)

MobileCount, we aim to increase the FPS value with a little decrease in accuracy.

Model	**Image Size**	**MAE**	**RMSE**
MobileCount(x0.5)	(1920, 1080)	25.201	36.608
	(960, 540)	41.636	59.587
MobileCount(x0.75)	(1920, 1080)	25.401	34.218
	(960, 540)(SGD)	29.880	38.882
MobileCount	(1920, 1080)	21.780	29.167
	(960, 540)(SGD)	30.287	38.428

TABLE IV: Comparison of the results of the various models

Model	**Image Size**	**Batch Size**	
		4	**8**
MobileCount	(1920, 1080)	15.38	16.01
	(960, 540)	34.44	35.70
MobileCount(x0.75)	(1920, 1080)	16.43	16.71
	(960, 540)	36.76	38.61
MobileCount(x0.5)	(1920, 1080)	19.21	19.20
	(960, 540)	40.62	41.37

TABLE V: Comparison of the results of the various models on the FPS at different batch sizes.

The experiment [Table V] showed how, when reducing the number of channels we can indeed obtain an increase in the FPS value; instead the batch size value seems to not have the expected effectiveness, and in some cases the FPS values even decreased. These values are strictly related to the used GPU and its parallel computational power, as a matter of fact the same test on a Tesla T4 did not gave significant difference when changing the different MobileCount versions.

After a more accurate observation, we noted that the test set was composed for a good part of night video sequences, sequences that in the train set were in short supply; in attempt to overcome this problem the random gamma correction described in IV-B was added to the data augmentation transformations so that darker images that simulate night time frames could be fed to the model during the training phase helping it to better estimate the number of people in the night time frames of the test set.

The results, as shown in Fig. 5 were different for every configuration of parameters; we still believe that a better studied data augmentation could reduce the estimation error of the models since where the result got worse it was only of a few points, except for MobileCount with full-sized images, which extremely good result we consider the product of an optimal random split of the train-validation set.

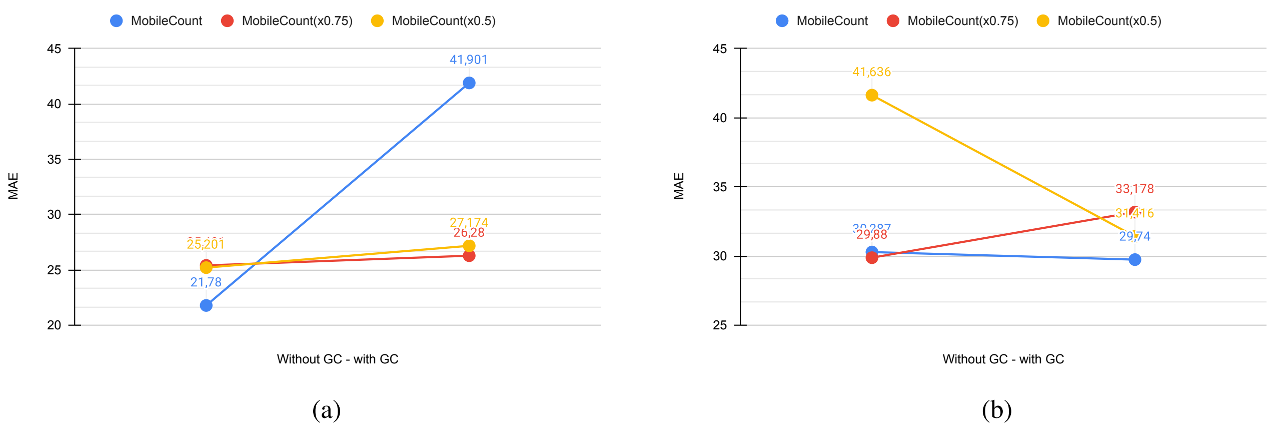

Fig. 5: Graphs that compare the Mae between the models without and with Gamma Correction in full-sized images (a) and half-sized images (b)

D. *State of the Art Comparison*

In this section we compare the best result achieved in this work [Table IV] with the current state-of-the-art counting approaches on the same dataset [Table VI].

Method	MAE	MSE
FPNCC	11.66	15.45
BVCC	12.36	17.32
CFF	13.64	16.59
PDCNN	13.79	21.50
DevaNetv2	15.45	19.90
CSRNet+	15.60	21.50
M-SFANet	15.83	21.18
SCNet	16.64	29.03
CSR-SSOF	19.70	27.69
MobileCount	21.78	29.17
Soft-CSRNET	26.85	37.79
CANet	26.98	38.50
MILLENIUM	51.63	68.56
SANet	57.47	65.35
ResNet-FPN	83.96	96.62

TABLE VI: Classification of the VisDrone challenge on crowd counting task

As we can clearly see, if submitted while the challenge was still open, would have classified between the best ten positions; this while also being a light weight model and thus being able to operate on mobile and embedded systems.

VII. Conclusion

Even thou the model has very good performances, as we can see in the Fig. 4 (f), for some problematic images, i.e. the night ones (Fig. 4 (g, h)), we get wrong heatmap estimation, and so wrong counts. Therefore more testing needs to be carried out in order to achieve even better metrics in the crowd counting estimation. Some future developments could be a better data augmentation in order to get better accuracy on night or dark images that are present only in a small quantity in the train set; another experiment that would be very useful could be the analysis of the performances in terms of FPS on a drone GPU or directly on an high-end drone. Another possible enhancement, is to use transfer learning from a pretrained mobile on a big dataset, such as ImageNet, in order to improve the first phase of encoding.

References

[1] Paul Viola, Michael J Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[2] Antoni B Chan and Nuno Vasconcelos. Bayesian poisson regression for crowd counting. In *2009 IEEE 12th international conference on computer vision*, pages 545–551. IEEE, 2009.

[3] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 833–841, 2015.

[4] Lokesh Boominathan, Srinivas SS Kruthiventi, and R Venkatesh Babu. Crowdnet: A deep convolutional network for dense crowd counting. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 640–644, 2016.

[5] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016.

[6] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4031–4039. IEEE, 2017.

[7] Deepak Babu Sam, Neeraj N Sajjan, R Venkatesh Babu, and Mukundhan Srinivasan. Divide and grow: Capturing huge diversity in crowd images with incrementally growing cnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3618–3626, 2018.

[8] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1091–1100, 2018.

[9] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–546, 2018.

[10] 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.

[11] 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.

[12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[13] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[14] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2017.

[15] Vladimir Nekrasov, Chunhua Shen, and Ian Reid. Light-weight refinenet for real-time semantic segmentation. *arXiv preprint arXiv:1810.03272*, 2018.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[17] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[18] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[19] Giovanna Castellano, Ciro Castiello, Marco Cianciotta, Corrado Mencar, and Gennaro Vessio. Multi-view convolutional network for crowd counting in drone-captured images. In Adrien Bartoli and Andrea Fusiello, editors, *Computer Vision – ECCV 2020 Workshops*, pages 588–603, Cham, 2020. Springer International Publishing.

[20] Junyu Gao, Qi Wang, and Xuelong Li. Pcc net: Perspective crowd counting via spatial convolutional network. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(10):3486–3498, 2019.

[21] Zan Shen, Yi Xu, Bingbing Ni, Minsi Wang, Jianguo Hu, and Xiaokang Yang. Crowd counting via adversarial cross-scale consistency pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5245–5254, 2018.

[22] Zenglin Shi, Le Zhang, Yun Liu, Xiaofeng Cao, Yangdong Ye, Ming-Ming Cheng, and Guoyan Zheng. Crowd counting with deep negative correlation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5382–5390, 2018.

[23] Peng Wang, Chenyu Gao, Yang Wang, Hui Li, and Ye Gao. Mobilecount: An efficient encoder-decoder framework for real-time crowd counting. *Neurocomputing*, 407:292–299, 2020.

[24] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Qinghua Hu, and Haibin Ling. Vision meets drones: Past, present and future. *CoRR*, abs/2001.06303, 2020.

[25] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

[26] Vishwanath A Sindagi and Vishal M Patel. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

Appendix

Here we can find the github repository with the code for reproduction of the experiment, and the weights of the trained models: CrowdCounting on VisDrone2020