CrowdCounting: An Attempt on VisDrone-2021 Challenge

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 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 be useful for a very large set of tasks e.g. gathering, detection. Research Questions: The main aim of this study is to find a 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 models taken from the Awesome Crowd Counting repository were examined and tested, then the models that were considered the best were fitted to the dataset. Findings: the tested models have very good performances in terms of counting, however with low-resolution resized images the obtained density maps are coarse.**

***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.

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.

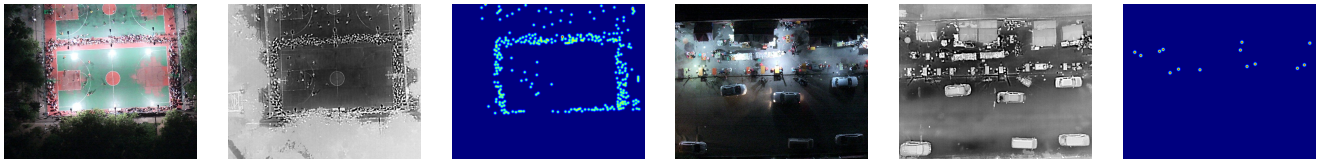
Regression-based method [2] were then developed; these methods directly computed the number of objects based on features extracted from cropped image patches. Despite the effectiveness in counting, these methods could not produce any other useful information on the input image.

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. [10] 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:

- PeopleFlow [11]: 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 [11], we were not able to train it from scratch as it requires an amount of VRAM that our setup could not handle;
- SASNet [12]: is a encoder-decoder net characterized by the use of both density maps and confidence maps, a reliability test was performed and the described performance were respected.
- MobileCount [13]: 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 [13] both in terms of accuracy and FPS.

The last two model were chosen to tackle this task, since both the two are very promising.



(a) (b) (c) (d) (e) (f)

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

III. The VisDrone Dataset

VisDrone [14] 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-Counting2021 dataset.

The VisDrone-Counting2021 dataset consists of 2723 frames, 1807 for training, 916 for testing. Each frame has a resolution of 640x512 pixel [Fig. 1]. For each frame is also provided an infrared picture of the same scene. 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 only use a simple random horizontal flip;

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 $p = 0.5$.

V. Method

The two chosen models share some similarities in their structure; they are both encoder-decoder organized in blocks: each block of the encoder propagate its output into the next encoder block and also into the corresponding decoder block, finally the decoder output is fed into a prediction block. In our work we exploited this structure trying different suitable pretrained encoders and combining the nets modules.

A. Base Neural Networks

1) MobileCount: MobileCount [13] is a Fully Convolutional Network composed of two parts: an encoder and a decoder. The encoder is based on the MobileNetV2 [15], 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. MobileCount reduces the number of bottlenecks from 7 to 4, a number obtained by some empirical study. The model is even lighter thanks to a 3×3 MaxPooling of stride 2 that reduce the resolutions for the next bottlenecks layers.

MobileCount, uses the Light-Weight RefineNet [16] decoder that was originally designed for semantic segmentation. This decoder has been specifically simplified from RefineNet [17] to achieve real-time performance, which can be combined with any encoder of multi-scale feature maps.

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. [Fig. 2]

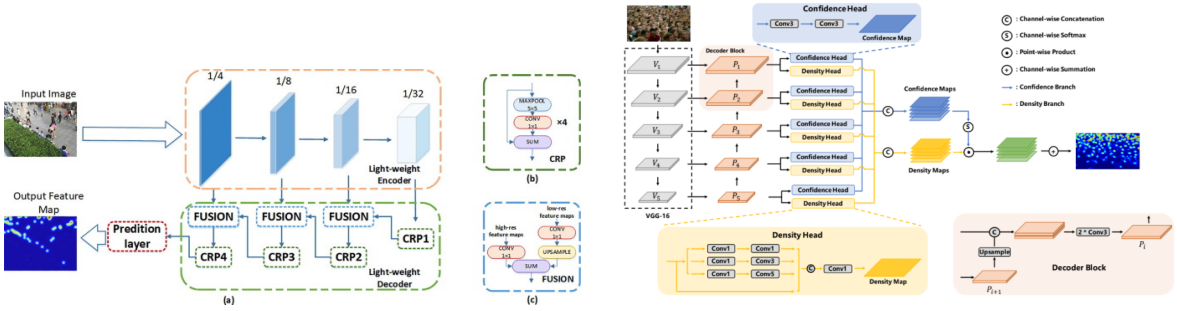
MobileCount offers 3 variants changing the number of feature channels, we added another two variants reducing even more the feature channels [Table I].

2) SASNet: is also a CNN composed of a backbone encoder (VGG-16), a decoder and a prediction block; the prediction block is composed of two parallel branches: density branch and confidence branch.

The density branch predicts density maps using five feature levels, it consists of five density heads which are responsible for the predictions using P1, P2, P3, P4, P5, which are generated hierarchically using the features of different levels in the back-bone network, respectively. The module consists of three convolution branches and one skip connection branch; for the convolution branches, firstly is used a 1×1 convolution for channel reduction, and then are used convolutions with

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 [13].



(a) (b)

Fig. 2: The MobileCount architecture (a) and SASNet architecture (b)

different kernel sizes to acquire context information from various receptive fields. The output features from the multi-branch module are concatenated along channel dimension, after which a 1 × 1 convolution allows to obtain the prediction for the i-th feature level. Finally, the predicted density map of each feature level is upsampled to the same size as original input.

The confidence branch is responsible for predicting confidence scores to indicate the most appropriate feature level for a certain image patch. After the predicted density maps D1, D2, D3, D4, D5 are acquired, a confidence branch is used to assist the feature level selection, which contains five confidence heads, one for each feature level. The confidence head predicts a score to indicate its confidence of being the most appropriate feature level for a certain patch. Confidence maps are denoted as C1, C2, C3, C4, C5, which are generated with the confidence heads using features P1, P2, P3, P4, P5. Specifically, first Pi is downsampled to 1/k size of original input image. Then the downsampled features are processed with two 3×3 convolutions, which are then modulated by a Sigmoid function to obtain Ci.

The final predicted density map is obtained using an adaptive selection strategy. [Fig. 2]

B. Encoders

MobileCount uses a custom Light-Weight encoder, SASNet uses VGG16; however we can found in literature better encoders, therefore we tried to train both the networks using different encoders: ResNet18, ResNet34, ResNet50 and InceptionNetV3.

1) Resnet: ResNet [18] is a special case of residual network. A residual network is a network in which, instead of approximating a mapping H(x), we let the network approximate a residual function F(x) := H(x) x. This formulation is used to address the degradation problem. With the network depth increasing, the accuracy gets saturated and then degrades rapidly. The ResNet architecture uses 3 × 3 convolutional layers with batch normalization and ends with a global average pooling layer and a fully-connected layer with softmax activation function. Shortcut connections are inserted. The identity shortcuts can be directly used when the input and output are of the same dimensions. dimension increase, an identity shortcut with 0-padding or a projection shortcut can be used. [Fig. 3]

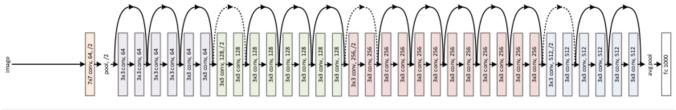


Fig. 3: The Resnet-34 architecture

2) InceptionNet v3: InceptionV3 [19] is a convolutional neural network which has some Inception modules. An inception module is a layer that performs multiple convolutions at the same level and returns a concatenation of all of them. The InceptionV3 architecture from the Inception family makes several improvements including using label smoothing, factorized 5 × 5 and 7 × 7 convolutions, and the use of auxiliary classifiers. Auxiliary Classifiers are type of architectural component that seek to improve the convergence of very deep networks. They are classifier heads we attach to

layers before the end of the network. The motivation is to push useful gradients to the lower layers to make them immediately useful and improve the convergence during training by fighting the vanishing gradient problem. [Fig. 4]

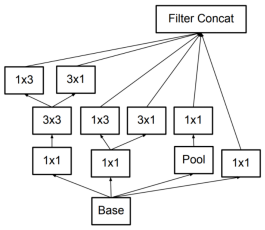


Fig. 4: The Inception v3 module architecture

C. Loss functions

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

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

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.

D. Thermal infrared (TIR) Adapting

Most of the pretrained network architecture take as input sizes the 3 RGB channels, but we have an extra infrared channel. So we decided to add an extra parallel encoder for the infrared channel and concatenated the output. We used both the MobileCount LightWeightEncoder and the pretrained network ResNet, InceptionNet and vgg16. For the last ones we repeated the infrared channel 3 times in order to adapt the input.

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 various configurations. Finally, we present the score obtained in the VisDrone Challenge.

A. Implementation Details

The training and testing are performed on Google Colab platform using a Nvidia Tesla T4 and PyTorch framework. In our experiment, we used both Adam optimization and SGD optimization (that has been early discarded due to poor results); the weight decay rate is set to $1 \times 10e-4$ while the initial learning rate is set to $1 \times 10e-4$.
Models have been validate using an holdout procedure. Validation/Test split hasn't been done randomly, that's because there are a lot of frames very similar, thus a random split could result in an optimistic error; in order to avoid this, we firstly grouped the frames into different video sequences, than extracted the 20% of the video sequences to get the test set. The process has been repeated to get also the validation set.

B. Evaluation Metrics

Following [3], [8], [13], [20], we use Mean Absolute Error (MAE) and Mean Square Error (MSE) to evaluate our model. They are defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|C_i^{PM} - C_i^{GT}| \tag{3}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(C_i^{PM} - C_i^{GT})^2} \tag{4}$$

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. The Loss problem

To better exploit the inference accuracy of the pretrained network, we resized the images to the size they were trained on, so 224x224 for Resnet and VGG16, and 299x299 for InceptionNet. A problem arose after the first experiments with low sized images: validation loss values went up exponentially reaching peaks of 600. This was due to the image being so small that there were not enough pixels to distinguish the different point representing the people, thus the upsampling produce a larger dense area instead of different dots [Fig. 6]. Density maps like these, however generate a good crowd count in terms of MAE and RMSE but they set doubts on the reliability of the loss measure. We made several attempts to fix this problem. Firstly we changed the upsampling method from a simple interpolation to a transposed convolution followed by a convolution, then we tried to increase the number of layers in output from the decoder and fuse them with a convolution after the upsampling. However none of these methods improved the results, because the information to distinguish the density dots is absent so there is no operation that can store it in its parameters; also there is no way to retrieve this features using the pretrained network architectures as encoders.

D. Comparison of the two nets

As first experiments, we trained from scratch and tested both MobileCount and SASNet and compared the obtained results, both on the full-size images and on the half-sized ones.

Model	Image Size	DM RMSE	MAE	RMSE
MC	(512x640)	15.21	11.71	18.57
SASNet	(224x224)	674.34	8.12	12.91

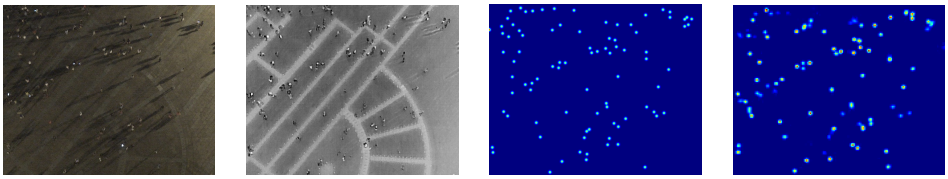
TABLE III: Comparison between the best results of MobileCount and SASNet. DM stands for Density Map

As showed in [TABLE III] both nets achieve good values of MAE and RMSE; mobile count also showed good RMSE on density maps and fast inference speed, SASNet on the other hand has a density map RMSE that is not comparable because of the problem explained in [Par. VI-C].

RGB Encoder	TIR Encoder	Decoder	Image Size	MAE	RMSE
ResNet 50	MC	MC	(512x640)	6.76	13.43
ResNet 34	MC	MC	(512x640)	11.45	18.11
ResNet 18	MC	MC	(512x640)	7.80	13.53
ResNet 50	ResNet 34	MC	(224x224)	8.80	15.80
ResNet 34	ResNet 34	MC	(224x224)	9.19	17.06
ResNet 34	ResNet 18	MC	(224x224)	10.86	17.46
ResNet 18	ResNet 18	MC	(512x640)	6.58	14.27
Inception v3	Inception v3	MC	(299x299)	7.43	13.49

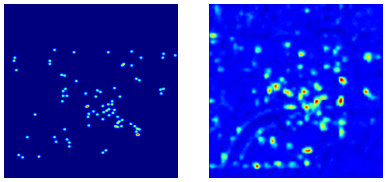
RGB Encoder	TIR Encoder	Decoder	Image Size	MAE	RMSE
ResNet 50	MC	SASNet	(512x640)	10.11	14.76
ResNet 34	MC	SASNet	(512x640)	8.47	17.65
ResNet 18	MC	SASNet	(512x640)	16.04	23.22
ResNet 50	ResNet 34	SASNet	(512x640)	10.06	15.57
ResNet 34	ResNet 34	SASNet	(512x640)	9.39	15.47
ResNet 34	ResNet 18	SASNet	(224x224)	8.84	16.70
ResNet 18	ResNet 18	SASNet	(512x640)	6.35	16.05
Inception v3	Inception v3	SASNet	too heavy, could not be trained		

TABLE II: Experiments done both on MobileCount and SASNet, reported best results between the two different sizes



(a) (b) (c) (d)

Fig. 5: A frames of the Visdrone dataset taken from out test set (a) with the respectively infrared image (b), generated heatmap (c) and predicted heatmap (d)



(a) (b)

Fig. 6: Density map ground truth (a). Predicted density map from SASNet with resnet18 as encoder with 224x224 resized images (b).

E. Combining Nets

In attempt to get a lower loss value, we tried to use transfer learning at our advantage by making use of a pre-trained net as encoder, such that we could have a better encoding and thus let the training process focus on the decoder.

The two nets that we tried as an encoder were ResNet and Inception v3; both of the models worked with different image sizes ((224x224) for the ResNet and (299x299) for the Inception v3) so an online resize of the images and a new generation of the ground truth was required.

As shown in [TABLE II] the models with two resnet18 obtained the best results, we suppose this represent the right capacity for this task with our architecture. We also think that is better to have a pretrained RGB encoder, and not pretrained TIR encoder since pretrained networks are trained on RGB channels.

F. Challenge Result

We choose 5 models based on the test set results, retrained them on the whole dataset and submitted the prediction on the official test set [Fig. 7]. The best model on our test set remains the same for the challenge; however our results metrics overestimate the obtained ones for the challenge. An explanation for this result could be the diversity between the challenge test set and the challenge training set from which our test and validation set were extracted, or our test set is not enough representative.

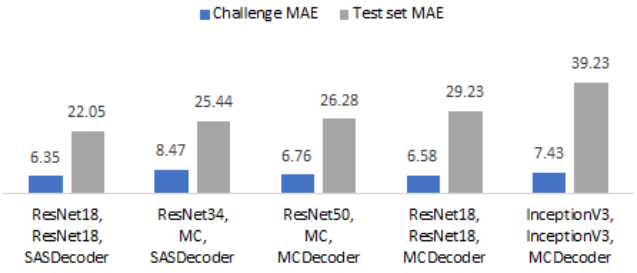


Fig. 7: Challenge vs. test results. (Encoder RGB, Encoder TIR, Decoder) on x-axis

VII. Conclusion

Models trained with full sized images reach very good perfomances in terms of counting and generate also accurate density maps [Fig. 5]. Models trained with downsampled images instead, achieve good performances in terms of counting but we obtain coarse density maps with low-resolution images. Therefore more testing needs to be carried out in order to achieve even better metrics in the crowd counting estimation and better density maps with low-resolution images. Some future developments could be:

- better data augmentation in order to increase the size of the training dataset;
- testing of the model on other datasets to have a comparison with the state in the crowdcounting field;
- analysis of the performances on a drone GPU or directly on an high-end drone;
- try another loss function to focus on the counting and exploit models trained on low resolution images.

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] 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.

[11] 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.

[12] Qingyu Song, Changan Wang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Jian Wu, and Jiayi Ma. To choose or to fuse? scale selection for crowd counting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(3):2576–2583, May 2021.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[20] 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.