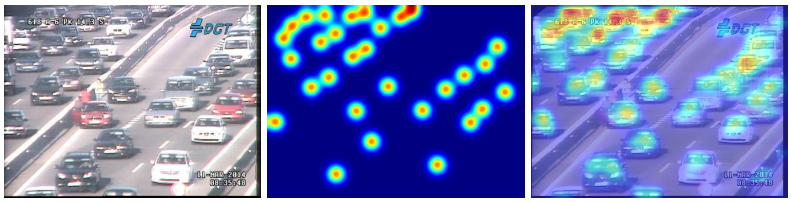
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2020
http://excel.fit.vutbr.cz

Counting Vehicles in Image and Video

Dominik Gabzdyl*



Abstract

Traffic analysis is still a challenging task. During such task there are many pitfalls to be aware of. Such as small image resolution, high number of overlapping objects, angle of camera, blurred objects due to their motion or weather conditions. This paper addresses the issue of counting vehicles instances in images and videos. Remarkable results and state-of-the-art methods are defined by convolutional neural networks. There are many approaches to address the issue of counting objects in images. One of which is counting by regression, which is the aim of this paper. I propose an architecture which is inspired by some state-of-the-art models. The proposed model improves accuracy on various datasets. For instance on the very small PUCPR+ dataset the Root Mean Square Error between expected and predicted vehicle counts was reduced from 34.46 to 8.84 vehicles (measured on the test set).

Keywords: vehicle counting — counting by regression — convolutional neural networks

Supplementary Material: *N/A*

*xgabzd00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Before the rise of Artificial Intelligence, vehicle detection task was carried out via hardware solutions. Inductive loops [1] solution, for instance, is one of them. Vehicle passing through such loop would cause change in magnetic field thus signalizing its presence. However such hardware solutions are costly. Recent major rise in computer vision presented approaches with high accuracy and low implementation cost.

In this paper I address the hard problem of precisely counting objects instances within an image (or video). The correct detection of an object is severely impacted by many factors. Some of which include extremely overlapping objects, distorted image, bad weather conditions, high density of objects, etc.

Data acquired from automated traffic flow analysis can improve management of traffic. Busy highways or road intersections can be monitored for traffic congestions to see how they evolve. Public authorities in charge of the maintenance and planning of road infrastructures can make use of such information.

2. Related work

Counting by regression belongs to supervised learning category. In supervised learning a ground truth (expected output) must be provided. One technique of the counting by regression approach is to use heatmaps. In this technique a pair of input image and its expected output heatmap image is provided to model. The training phase of model is then basically learning mapping between ground truth images and input images.

Following the idea of Lempitsky et al. [2] the counting problem is then a process of recovering a density function as a real function of pixels of input images. Predicted heatmap patch of objects is the output from a network of this approach. Expected

heatmap patch is made by applying Gaussian blur with a reasonable standard deviation value σ (in this work $\sigma = 15$) centred on each dot in the annotated image.

However, this will get us a full ground truth image. We need to split input images and ground truth images into patches. Note that each input and ground truth patch pair must have matching coordinates (their patch coordinates must correspond).

Gaussian blur uses a kernel whose values add up to 1. This ensures no energy is added or removed from the image after this operation. Thus, the sum of convolved dots (pixels) in the annotated image is unchanged (the vehicle count is preserved). To get the final count we simply sum up the density map.

In this section I will also discuss architectures which inspired my work. I will provide a brief summary of each. I will also mention pros and cons where possible.

2.1 Multi-column Deep Neural Networks for Image Classification

This network architecture [3] is inspired by vertical column-like arrays of neurons (located in the temporal cortex of human).

The model is made of vertically stacked columns to learn image features independently and in parallel. Each column consists of the same combination of convolutions, pooling layers and fully connected layers. An image gets a unique distortion/preprocessing before it is passed into a column. At the end all columns are averaged and a final output is produced.

2.2 Counting CNN

One of the proposals of the paper *Towards perspective-free object counting with deep learning* [4] is the Counting CNN. The model falls into counting by regression category, which is the category covered in this paper. In the training phase 800 RGB patches are randomly cropped with a fixed size from an input image and are fed to the model. Augmentation strategy is done by vertical flipping each patch making it 1,600 patches in total. They crop patches of size 72x72x3 and 18x18x1 patch is the direct output from their network. The prediction is done via dense patch extraction and the output patches are assembled together to generate the final ground truth estimation. Counting CNN was also trained on the TRANCOS dataset (see TRANCOS dataset) and reported an improved accuracy on this dataset [4].

Their architecture is rather small than large (6 convolutional layers in total). This makes it relatively fast to predict an output patch from the input one. If the number of training patches is smaller, training will be a lot faster with not so big accuracy impact.

2.3 Context-Aware Crowd Counting Network

Even if the Counting CNN tries to solve the perspective distortion by random patch extraction, its results indicate severe inaccuracies since the scale continuously varies across the whole image [5]. The Context-Aware Crowd Counting Network (CAN) adaptively encodes multi-level contextual information into features it produces. They perform so-called Spatial pyramid pooling [6]. Scale-aware features are computed by average-pooling subtraction from the VGG-16 [7] front-end. The front-end uses a method known as transfer learning. This means pre-trained weights are downloaded and used instead of initializing weights by some distribution, thus significantly reducing the training time. CAN model downloads pre-trained weights from the ImageNet dataset [8] and adjusts (trains) them to fit a particular dataset. In the training phase input image is divided into four regions of equal size (non-overlapping). One of these four regions is randomly chosen and passed to the net.

2.4 Pyramid Density-aware Attention Net for Accurate Crowd Counting

A very recent architecture (January 2020) to the time of writing my thesis [9]. PDANet continues to improve the accuracy of the CAN model.

This model also makes use of VGG-16 front-end for low-level feature extraction. Pyramid feature extraction with spatial and channel attentions are attached to the front-end to produce richer features. The model distinguishes between images with sparse and dense object instances. To distinguish between these two a classification module is used which then decides whether the image is either sparse or dense.

Within the decoder module there are two branches: dense and sparse. The model is trained in a such way that weights in the upper branch (dense) are adjusted for dense features and the lower branch (non-dense) is adjusted for sparse ones. At the end these branches are combined together to produce the final density estimation.

3. Proposed solution

The proposed architecture makes use of VGG-16 front-end for low-level features extraction. The CAN and PDANet use only first 10 layers of VGG-16 and so does the proposed solution. The architecture is depicted in the Fig. 1. The front-end is then connected

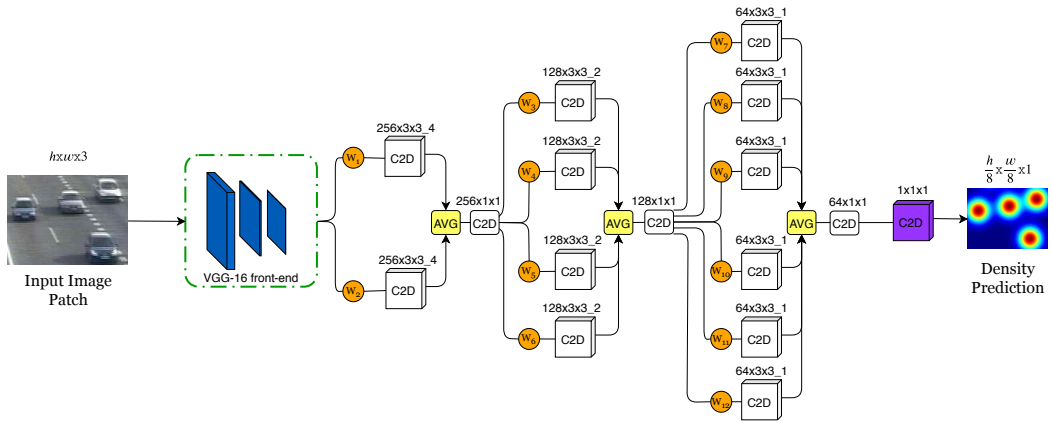


Figure 1. Proposed architecture. RGB patches are fed to the VGG-16 front-end network. Resulting low-level features are weighted (neurons w_i) and sent to dilated convolutions in parallel. Averaging followed by 1x1 convolving ends one such parallel layer. Dilated convolutions in parallel layers increase the receptive field of the network and help to understand the overall picture. The last parallel layer gets additional 1x1 convolving and produces density estimation. Input patch height and width dimensions are reduced by a factor of 8 due to two max-pooling layers in the VGG-16 front-end.

Table 1. Dilated convolutions setup

Structure number	Filter depth	Dilatation rate
1.	256	4
2.	128	2
3.	64	1

to a tree-like structure, which is inspired by aforementioned Multi-column network.

One layer (structure) of the entire tree-like structure is composed as follows. Input to this layer gets weighted n-times (where n is the number of parallel conv layers within the corresponding structure). Each weight w_i represents one neuron, whose value is determined during the training phase. These weights use Softmax activation functions. Initially the value of each weight w_i is set to an increasing multiple of $1/n$. So for example the last structure has the initial values $w_7 = 1/6$, $w_8 = 2/6$, $w_9 = 3/6$, $w_{10} = 4/6$, etc.

Each weight is connected to a dilated conv layer (C2D white box in the Fig. 1). All dilated convolutions use the same kernel size 3x3 but they differ in dilatation rate and filter depth. The setup of these conv layers is shown in the Table 1 (structure count starts from the front-end).

Dilated convolutions are then averaged. Final 1x1 convolving ends one such structure. All convolutions in each structure are followed by ReLU activation functions.

The final structure gets additional 1x1 convolution to produce a density estimation for a given input patch.

The weighting idea came from PDANet classifier, which is responsible for giving each of the two branches unique weights. Dilated convolutions increase the receptive field of the network and help to understand the overall picture instead of finer details [10].

I have implemented this architecture in Keras framework [11]. The model uses Mean Square Error (MSE) loss and Adam optimizer to compute weights.

Training setup Input RGB patch extraction is done via division of input image into four equal regions and randomly picking one of them. This patch gets randomly chosen combination of augmentation (zoom, horizontal flipping, gamma change, gaussian noise addition). The front-end has 2 max-pooling layers, which reduce input patch dimensions by a factor of 8. Therefore, the corresponding ground truth patch needs to be scaled down. Simply stretching or shrinking the patch could change the count (hence patch normalization is needed). In the *Towards perspective-free object counting with deep learning* paper [4] a normalization formula was proposed to solve such issue (see Equation 1).

$$\hat{D}^{P+}_{pred} = \frac{\sum_{\forall p} D^{P}{}_{pred}(p)}{\sum_{\forall p} \hat{D}^{P}_{pred}(p)} * \hat{D}^{P}_{pred} \qquad (1)$$

where:

- $\hat{D}^{P+}_{pred}$: is the rescaled density patch *with* a matching count
- $D^{P}{}_{pred}$: is the original (not rescaled) density patch
- $\hat{D}^{P}_{pred}$: is the rescaled density patch *without* a matching count
- p: is the pixel

Dense patch extraction is a chosen method to produce final density estimation (each patch is a quarter of an image). Since patches are overlapping the final

Figure 2. Sample images from specified datasets. **Top row:** TRANCOS **Bottom left:** PUCPR+ **Bottom right:** CARPK

density map must be normalized. Each position (pixel) in the final density map is an average of patches that cast a prediction in it.

4. Comparison with state of the art

The proposed architecture is trained and tested on the following datasets. Each dataset will be introduced and results achieved will be shown in tables. I will compare official state-of-the-art results with my model. All official state-of-the-art results are pulled from the respective dataset web page. Tables with comparisons are truncated to show only Top 5. Values in these tables depict error rates between expected and predicted vehicles counts measured on the corresponding test sets (the lower the value the more precise the prediction is).

4.1 TRANCOS dataset

A Spain car dataset [12]. TRANCOS is the acronym for TRaffic ANd COngestionS. As the name suggests the dataset focuses on traffic congestions. The dataset was made by capturing traffic via selected real surveillance cameras during three week period. The dataset contains more than 1,200 images and 46,700 annotated vehicles. These images are split in 3 sets as follows:

- Training - 403 images
- Validation - 420 images
- Test - 421 images

Since annotations are partial (not every vehicle is annotated), they used Region of Interest (ROI). Binary masks are provided to make visible only the annotated section of image (leaving non-ROI regions black). To train a model we are given a choice between 2 methods: either train solely on training set or to train on both training and validation sets (or as they call it - 'trainval' set).

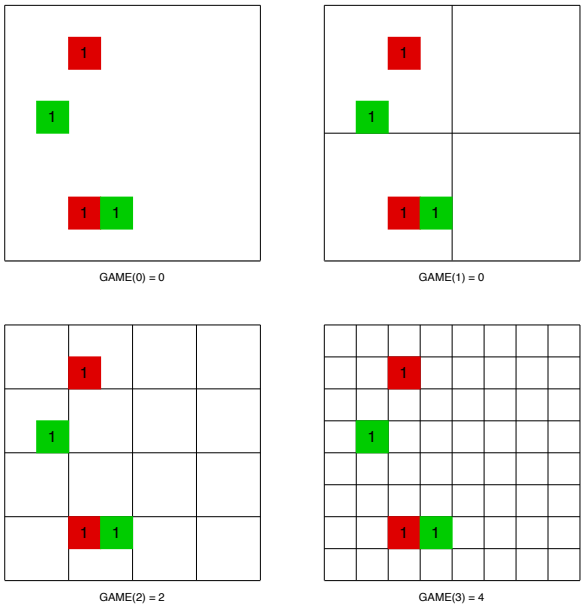


Figure 3. GAME(L) metric visualization

TRANCOS uses its own metric to measure network's accuracy – the $GAME(L)$ metric. $GAME(L)$ stands for Grid Average Mean Error and is required to be used when reporting results of one's model accuracy on this dataset. State-of-the-art results are shown in the Table 2. The $GAME(L)$ metric is formulated in the Equation 2.

$$GAME(L) = \frac{1}{N} \cdot \sum_{n=1}^{N} \left(\sum_{l=1}^{4^L} \left| e_n^l - gt_n^l \right| \right), \qquad (2)$$

where:

- N: is the total number of test images
- L: is the total levels number (max 4-regions exponent)
- e_n^l: is the estimated count within region l of image n
- gt_n^l: is the ground truth count within region l of image n

This metric takes into consideration vehicle localization error. The higher the L number the more strict /precise localization error estimation is. I have made a toy example for demonstration purposes (see Fig. 3). In this figure there are red and green boxes with numbers within a grid. Red boxes represent expected vehicles within a region, whilst green ones represent vehicles prediction within a region. TRANCOS has matlab code for calculating this metric (publicly available along with the dataset on their web page [17]). I have rewritten this code from matlab to python to get the $GAME(L)$ evaluation of my model which is written in Keras [11].

Table 2. Evaluation of state-of-the-art methods on dataset TRANCOS. Lower number is better.

Method	GAME(0)	GAME(1)	GAME(2)	GAME(3)
IbPRIA 2015 (HOG-2) [12]	13.29	18.05	23.65	28.41
IbPRIA 2015 (Fiaschi+RGB Norm+Filters) [12]	17.68	19.97	23.54	25.84
IbPRIA 2015 (Lempitsky+SIFT) [12]	13.76	16.72	20.72	24.36
Hydra CNN (ECCV2016) [4]	10.99	13.75	16.69	19.32
proposed model	**3.08**	**4.29**	**5.97**	**7.95**

Table 3. Evaluation of state-of-the-art methods on dataset PUCPR+. Lower number is better.

Method	MAE	RMSE
YOLO [13]	156.00	200.42
Faster R-CNN [14]	111.40	149.35
One-Look Regression [15]	21.88	36.73
Layout Proposal Network [16]	22.76	34.46
proposed model	**7.48**	**8.84**

Table 4. Evaluation of state-of-the-art methods on dataset CARPK. Lower number is better.

Method	MAE	RMSE
YOLO [13]	48.89	57.55
Faster R-CNN [14]	47.45	57.39
One-Look Regression [15]	59.46	66.84
Layout Proposal Network [16]	23.80	36.79
proposed model	**15.62**	**18.43**

4.2 PUCPR+ dataset

This dataset is a subset of the PKLot dataset [18]. As per description the captures are taken at Pontifical Catholic University of Parana (PUCPR), located in Curitiba, Brazil. Images were taken from the 10th floor of the administration building of the PUCPR. These captures were taken during various weather conditions such as sunny, overcast and rainy.

Originally, the PUCPR dataset (note the missing '+' sign) is annotated only partly (100/331 parking spaces in a single image). Meng-Ru et al. [16] localized and completed missing annotations and called it PUCPR+. In total of only 125 images with nearly 17,000 cars, the 25 of them are meant for model testing. Such low number of images makes it a challenging task to train a network. State-of-the-art results are shown in the Table 3 (*MAE* and *RMSE* stand for Mean Absolute Error and Root Mean Square Error respectively). The dataset is available on the github [19].

4.3 CARPK dataset

Meng-Ru et al. proposed their own dataset and called it CARPK [16]. CARPK provides large-scale images taken from the drone point of view. The drone took images from four various parking lots. There are nearly 1,500 images with approximately 90,000 cars in total. Drone addresses the problem of a bias created by a fixed camera scene, where the point of view is constant. Images annotations are done by bounding box technique, where each vehicle gets upper left and bottom right (x, y) coordinates representing the bounding box. For the purpose of my regression approach this bounding box annotation is converted to dot annotation by taking a centre of the bounding box. State-of-the-art results are shown in the Table 4 (*MAE* and *RMSE* stand for Mean Absolute Error and Root Mean Square Error respectively). As well as the PUCPR+ dataset this one is also available on the github [19].

5. Conclusion

The paper presented a neural network architecture for counting vehicles instances in images. Proposed architecture is based on the VGG-16 front-end and multi-column structures setup, which experimentally improved current state-of-the-art results on datasets mentioned in the Section 4. To summarize, the contributions made in this paper are as follows.

Firstly, introduction of the new neural network architecture for counting vehicles, which is inspired by state-of-the-art architectures.

Secondly, evaluation of the architecture on vehicle datasets and comparison with current state-of-the-art architectures.

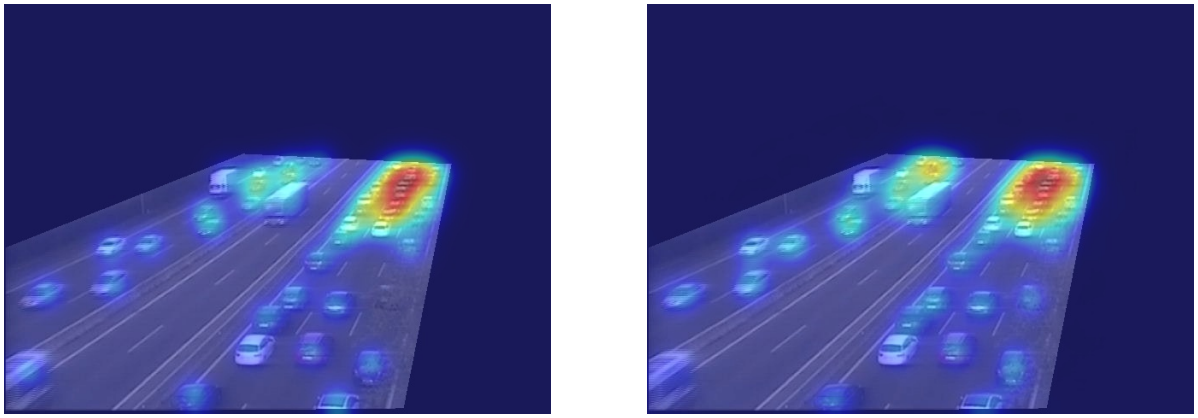
Lastly, further improvement of accuracy of state-of-the-art results on vehicle datasets. Some of the network predictions are shown in Figures: 4, 5, 6, 7. However, there is still space for improvements and possible further research in counting by regression approach.

Acknowledgements

I would like to thank my supervisor Ing. Jakub Špaňhel for his patient guidance, encouragement and advice he has provided throughout writing this paper.

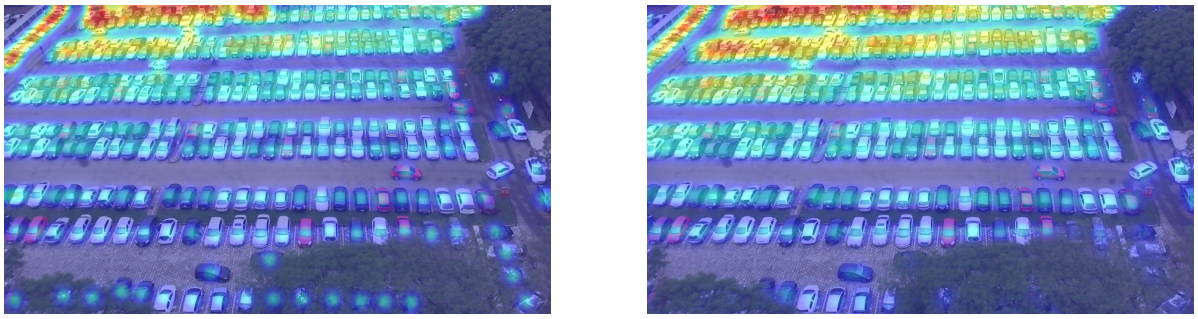
References

[1] Lawrence A. Klein, Milton K. Mills, and David R. P. Gibson. *Traffic detector handbook*, volume 1, chapter 2. 3 edition, Oct 2006. Tech Report.



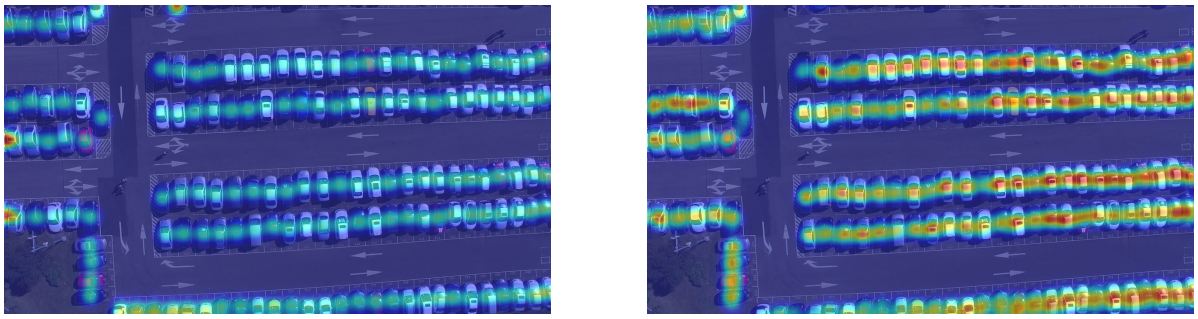
(a) Ground Truth: 44.03 **(b)** Prediction: 41.89

Figure 4. TRANCOS prediction



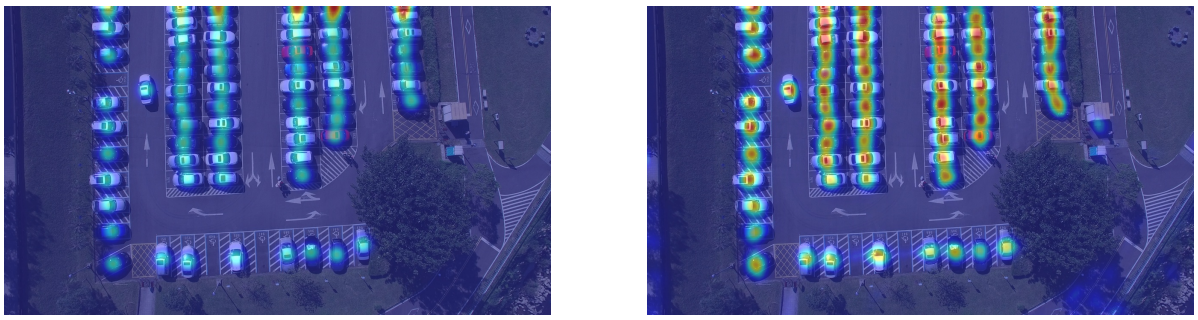
(a) Ground Truth: 328.77 **(b)** Prediction: 328.10

Figure 5. PUCPR+ prediction



(a) Ground Truth: 152.41 **(b)** Prediction: 143.76

Figure 6. CARPK prediction



(a) Ground Truth: 67.15 **(b)** Prediction: 63.94

Figure 7. CARPK prediction

[2] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. pages 1324–1332, 01 2010.

[3] Dan Cireşan, Ueli Meier, and Juergen Schmidhuber. Multi-column deep neural networks for image classification, 2012.

[4] Daniel Oñoro and Roberto López-Sastre. Towards perspective-free object counting with deep learning. volume 9911, 10 2016.

[5] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[8] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[9] Saeed Amirgholipour, Xiangjian He, Wenjing Jia, Dadong Wang, and Lei Liu. Pdanet: Pyramid density-aware attention net for accurate crowd counting, 2020.

[10] mc.ai. *10. Introduction to Deep Learning with Computer Vision - Types of Convolutions & Atrous Convolutions*. https://mc.ai/10-introduction-to-deep-learning-with-computer-%vision-types-of-convolutions-atrous-convolutions-2. [Online; visited 12.03.2020].

[11] Keras. *Keras Documentation - Home*. https://keras.io/. [Online; visited 31.01.2020].

[12] Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez, Roberto López-Sastre, Saturnino Maldonado-Bascón, and Daniel Oñoro. Extremely overlapping vehicle counting. 06 2015.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[15] T. Nathan Mundhenk, Goran Konjevod, Wesam A. Sakla, and Kofi Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 785–800, Cham, 2016. Springer International Publishing.

[16] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H. Hsu. Drone-based object counting by spatially regularized regional proposal networks. In *The IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[17] TRANCOS. *TRANCOS dataset - Download*. http://agamenon.tsc.uah.es/Personales/rlopez/data/trancos/TRANCOS_v3.tar.gz. [Online; visited 31.01.2020].

[18] Paulo Almeida, Luiz Soares de Oliveira, Alceu Jr, Eunelson Jr, and Alessandro Koerich. Pklot - a robust dataset for parking lot classification. *Expert Systems with Applications*, 42, 02 2015.

[19] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H. Hsu. *CARPK, PUCPR+ datasets - Download*. https://lafi.github.io/LPN/. [Online; visited 31.01.2020].