Regression vs. Density-Based Crowd Counting: Mall Dataset Case Study

Dejan Dichoski
dejan.dichoski@studenti.unipd.it

Süleyman Erim
sueleyman.erim@studenti.unipd.it

Maksim Kokot
maksim.kokot@studenti.unipd.it

Abstract

In this study, we address the crowd counting problem, by comparing regression-based (Xception) and density-based (CSRNet) approaches, and focussing the inference on the Mall dataset. Through various experiments, including hyperparameter optimization and transfer learning, the Xception model, emerges as the top performer, yielding exceptional results on the test set with an MAE of 1.5996. On the other hand, the CSRNet architecture achieves weaker performance with MAE of 3.44 when using a pretrained model and 2.72 when used as a feature extractor.

1. Introduction

This study delves into the challenge of crowd counting, an important task with direct implications for crowd control and public safety. Accurate crowd counting is vital for various applications, including assessing the impact of political campaigns, analyzing crowd behavior, ensuring the safety of drone landings, and preventing the spread of viruses.

In this context, our research explores two distinct approaches to crowd counting: a regression-based method and a density-based method. The first approach directly estimates the number of people by performing regression with a deep CNN, such as the Xception model. The second approach performs an indirect estimate of the number of people. First, the density of people in the image is estimated utilizing the CSRNet architecture [12], and then the count is inferred starting from the obtained density map.

Through in-depth experimentation and analysis, we present noteworthy insights and promising results. Our regression-based approach, particularly with the Xception model [3], showcases the effectiveness of fine-tuning pre-trained models, achieving a remarkable MAE of 1.5996 on the Mall dataset. Furthermore, our density-based approach, employing a pre-trained CSRNet, achieves an MAE of 3.44, and 2.72 when the CSRNet architecture is used as a feature extractor for a regressor network.

Thus, a simple regression based on neural networks could perform as well as a density-based approach, likely because density estimation methods are better suited for dense datasets, unlike the Mall dataset, which is somewhat sparse. These findings offer valuable contributions to the field of crowd counting, providing a foundation for further exploration and improvement in diverse scenarios.

2. Related Work

Crowd counting has been a prominent research area with various methodologies proposed in the literature. Here we highlight the most important ones:

- Detection-based approaches: Early research emphasized moving-window-like detectors [6], utilizing classifiers like Haar wavelets and HOG [17, 4]. Challenges arose in crowded scenes, leading to modifications like detecting specific body parts [7].

- Regression-based approaches: To address challenges in crowded scenes, researchers turned to regression-based approaches, learning relations among features to calculate object counts [7]. Idrees *et al.* [10] incorporated Fourier analysis and SIFT for interest-point-based counting.

- Density estimation-based approaches: Solving issues overlooked in regression, methods like that of Lempitsky *et al.* [11] considered saliency in local regions. Pham *et al.* [13] employed random forest regression for a non-linear mapping.

- CNN-based Approaches: CNN's have shown superiority in predicting density maps, outperforming previous methods [3].

In our work, we employ two of them, specifically density estimation-based, using CSRNet, and CNN-based using the Xception architecture.

3. Dataset

3.1. Mall Dataset

Chen *et al.* [2] compiled this dataset comprising 2,000 consequential images of pedestrians in a mall, captured using a fixed camera. Each image has a resolution of 480×640 pixels (height and width) and showcases pedestrians navigating a crowded mall (see Figure 1).

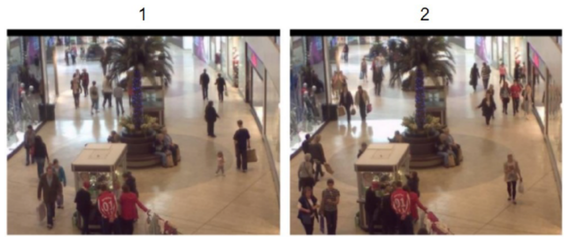

Figure 1: Two samples of Mall Dataset Images

We used this dataset for both training and testing. The dataset was divided randomly into trainval and test sets using a fixed random seed (42). Visual assessment confirmed that both sets originate from the same distribution. Consequently, as depicted in Figure 2, the trainval and test sets adequately represent crowd sizes ranging from approximately 20 to 50 people, with an average around 30. The test/trainval split corresponds to 20% and 80% (400 and 1600 images, respectively). Throughout the training process, the trainval dataset undergoes further division into train and validation images at proportions of 80% and 20% (1280 and 320 images, respectively).

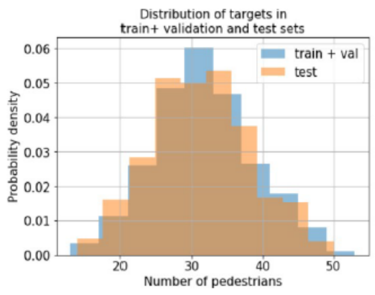

Figure 2: Distribution of target variable (number of pedestrians) in the trainval and test sets

3.2. Shanghaitech Dataset

This is a comprehensive crowd counting dataset introduced in the MCNNN paper [19]. It contains 1198 annotated images, featuring a total of 330,165 individuals with their head centers annotated. It is divided into two distinct parts: Part A, consisting of 482 images randomly sourced from the Internet, and Part B, encompassing 716 images captured from bustling streets in metropolitan areas within Shanghai.

Figure 3 illustrates the crowd histograms of images within this dataset, and Figure 4 illustrates the sample images. Part A is extremely dense, while Part B is sparser, but it is still a dense dataset.

We solely used this dataset for pre-training the Xception model, not for inference.

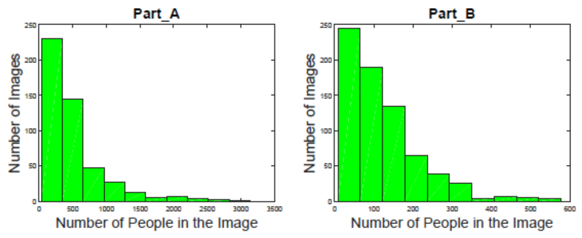

Figure 3: Crowd histogram of Shanghai Tech Dataset

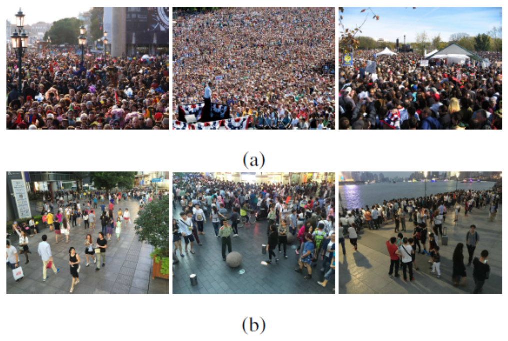

Figure 4: Shanghai Dataset (a) Representative images of Part A. (b) Representative images of Part B.

4. Method

We have proposed two different approaches to solve the crowd counting problem. Both of them are explained in the subsections below. Each approach was evaluated and compared via mean absolute error (MAE), mean squared error (MSE), and mean absolute percentage error (MAPE), whose formulas we report below:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (1)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (2)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \quad (3)$$

Where:

- n is the number of samples,
- y_i is the actual value for sample i,
- $\hat{y}_i$ is the predicted value for sample i,
- $|\cdot|$ denotes the absolute value, and
- The sum is taken over all samples.

MSE and MAE are sensitive to the scale of the data since they use the errors directly, not percentage errors. But in our case, all the inputs were scaled accordingly to the mean and standard deviation of the ImageNet dataset, so we didn't have this problem.

Additionally, MSE penalizes larger errors more heavily than smaller ones since it uses a squared difference. MAE and MAPE are more robust to outliers.

4.1. Regression Based Approach

We propose a regression-based approach to estimate the number of people in crowded scenes. Our initial strategy involves leveraging pre-trained deep CNN's as feature extractor on the Mall dataset images (Models: Inception v3 [16], Inception ResNet V2 [15], VGG16 [14], and VGG19 [14], ResNet50 [9], Xception [3]). Specifically, we employ each CNN with ImageNet learned weights, keeping them frozen during training to utilize them as a powerful feature extractor. Central to the success of this approach is the careful preprocessing of Mall dataset images, mirroring the preprocessing steps applied to the ImageNet dataset images prior to training the indicated models' networks.

After choosing the best model, we can proceed with identifying a promising baseline. We follow the methodology outlined by Geron (2022) in investigating the generalization ability based on the number of unfrozen layers. By keeping certain layers fixed, our approach allows us to leverage fine-tuned weights while also conserving computational resources.

Our method comprises the following steps:

- Base Model Selection: Choose a suitable base model for regression-based estimation.
- Selecting Number of Layers to Unfreeze: Investigate the impact of unfreezing varying numbers of layers on generalization.
- Adding Data Augmentation and Model Complexity: Enhance the model's ability to generalize by incorporating data augmentation techniques and increasing model complexity with additional layers.
- Activation Function Selection: Explore and select appropriate activation functions for the regression task.
- Two-Step Fine-Tuning for Better Weight Initialization: Fine-tune the model in two steps, by pre-training additional layers, then post training additional part with unfrozen layers again to optimize for weight initialization and convergence.
- Pre-training with Shanghai Tech Dataset: Utilize the best base model for pre-training on Shanghai Tech Dataset by unfreezing different number of layers for pre-training and post-training. Use different metrics to evaluate best approach (MSE, MAPE).

By adopting these comprehensive steps, we aim to optimize the regression-based estimation of crowd sizes, leveraging the strengths of pre-trained deep CNNs and fine-tuning strategies for specific dataset characteristics.

4.2. Density Based Approach

Our second approach performs an indirect estimate of the number of people. First, the density of people in the image is estimated, and then, starting from the obtained density map, the count is inferred [12].

As a first step, since we want the model to estimate the crowd density, we need ground truth density maps, i.e., targets. A crowd density map is a type of image label that can reflect the distribution of crowd heads by processing the head coordinate value through Gaussian convolution. So, we need head annotations, which can be seen as sparse matrices, whose dimensions are the same as the image, with all zeros but one entry equal to one, corresponding to the pixel in the center of the head.

Following the method of generating density maps in [12], we use geometry-adaptive kernels. The density map F is obtained via the following formula:

$$F(x) = \sum_{i=1}^{N} \delta(x - x_i) * G_{\sigma_i}(x) \text{ with } \sigma_i = \beta \bar{d^i} \quad (4)$$

For each targeted object x_i in the ground truth δ, we use $\bar{d^i}$ to indicate the average distance of k nearest neighbors. To generate the density map, we convolve $\delta(x - x_i)$ with a Gaussian kernel with parameter σ_i (standard deviation), where x is the position of the pixel in the image. $\delta(\cdot)$ is the discrete version of the delta function: this function is zero on all possible points but in zero. In our case, $\delta(0) = 1$.

We followed the configuration in [12], where β = 0.3 and k = 3. Finally, because the input is a sparse crowd, we adapt the Gaussian kernel to the average head size to blur all the annotations.

The CSRNet architecture that we utilized is the one that performed the best in the experiments of [12] - shown on Figure 5. This network is composed of two major components: a convolutional neural network (CNN) as the front-end for 2D feature extraction and a dilated CNN for the back-end, which uses dilated kernels to deliver larger reception fields and to replace pooling operations.

input(unfixed-resolution color image)
front-end (fine-tuned from VGG-16)
conv3-64-1 conv3-64-1
max-pooling
conv3-128-1 conv3-128-1
max-pooling
conv3-256-1 conv3-256-1 conv3-256-1
max-pooling
conv3-512-1 conv3-512-1 conv3-512-1
back-end
conv3-512-2 conv3-512-2 conv3-512-2 conv3-256-2 conv3-128-2 conv3-64-2
conv1-1-1

Figure 5: CSRnet. The conv layers' parameters are denoted as "kernel size - number of filters - dilation rate".

The CNN used as a front-end is VGG-16 pretrained on ImageNet, with the last 6 layers (fully-connected layers) removed (See Figure 6). Similarly to our previous approach, the input images were standardized using the mean and standard deviation of the ImageNet dataset. The output size of this front-end network is $\frac{1}{8}$ of the original input size, due to the three MaxPooling operations.

The back-end utilizes dilated convolutional layers for extracting deeper information of saliency as well as maintaining the output resolution. Dilated convolution uses sparse kernels to alternate the pooling and convolutional layer. This character enlarges the receptive field without increasing the number of parameters or the amount of computation. [12] For these layers, the initial values come from a Gaussian initialization with a 0.01 standard deviation.

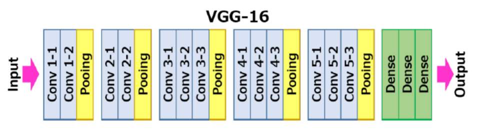

Figure 1: VGG16 deep CNN architecture.

Figure 6: VGG16 CNN model architecture

So, after the image passes through the whole network, it's going to be reduced to just $\frac{1}{8}$ of its original size. Therefore, we modified our targets to match this size, using bilinear interpolation with a factor of 8 for scaling.

5. Experiments

All of the performed experiments are available via the GitHub link [5].

5.1. Regression Based Approach

We compared models: Inception v3, Inception ResNet V2, VGG16, VGG19, ResNet50, and Xception with a full pipeline for crowd counting, using the Adam optimizer and a batch size of 64. Each model was trained for 50 epochs, employing a learning rate scheduler and early stopping to track validation MAE.

Xception was identified as the best model for further investigation (MAE: 2.6532), as it exhibited the lowest validation MAE, and its validation-loss curves appeared smooth without any signs of overfitting. Additionally, it can be observed from Figure 7 that the Xception model demonstrated faster convergence compared to other models.

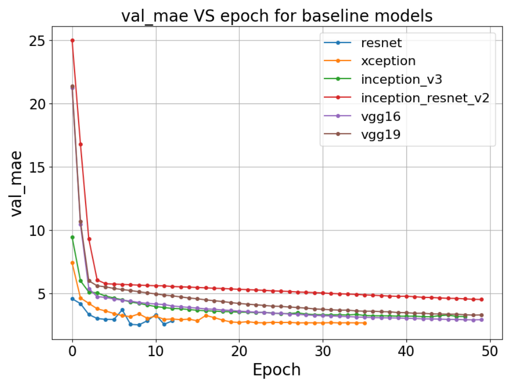

Figure 7: Base model validation curves

Then, various last layers of the Xception net are unfrozen to choose the optimal number of layers for fine-tuning the network. We experimented with 10-15-20-25-30-40 layers, respectively, and found that 15 (Figure: 8) is the best number of layers, yielding a lower validation MAE of 1.8758.

In the next scenario, we introduced data augmentation and additional layers to increase the complexity of the training data and enhance the model's complexity. This approach aims to improve the generalizability of our model across various cases while considering the bias-variance trade-off.

We used random horizontal flip as augmentation and 3 dense layers, each with 400 neurons, accompanied by dropout in between and ReLU activation functions. The model underwent training for 50 epochs with early stopping and a learning rate scheduler (ReduceLROnPlateau) to regulate validation MAE. The results improved from the baseline (MAE: 2.6532) to data augmentation (MAE: 1.77). However, the addition of extra layers with ReLU did not yield the expected outcome, as the validation MAE in-

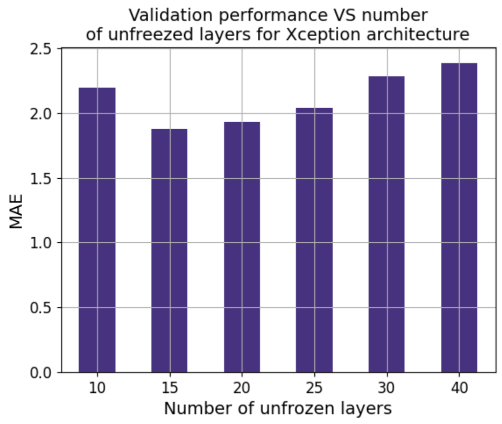

Figure 8: Xception unfrozen layers results

creased (2.2872). Subsequently, we proceeded with the Elu activation function without additional layers and observed an improvement in validation MAE (1.9273).

Even though the result was improved with Elu activation, it did not surpass the result obtained without additional layers. Therefore, the inclusion of extra layers did not perform as anticipated. Further experimentation was required, including exploring better weight initialization and adjusting the learning rate.

Following [8], the pre-training procedure was implemented for better weight initialization of additional layers. Different numbers of pre-training epochs were used, followed by 50 post-training epochs with various learning rates. The best result (MAE: 1.5996) was achieved through the following approach: firstly, only the additional layers were trained for 5 epochs with a learning rate of 0.001, and then, 15 layers from the original Xception, along with the additional layers, were unfrozen and trained for 50 epochs with a learning rate of 0.001, incorporating a learning rate scheduler (ReduceLROnPlateau) and early stopping. The summary of results is reported in Table 1.

Additionally, we tested a pre-training approach that involves another dataset designed for the same task. Here we used the same pipeline (Xception + 3 dense layers) as before. For this project, we used the Shanghaitech dataset during the pre-training phase. The pre-training algorithm is mainly based on considerations described in [8]. While conducting this step, we encountered several challenges. First, the Mall and Shanghaitech datasets had different distributions of the target variable. That is, more than half of the instances in the Shanghaitech dataset had a number of people greater than the maximum number of people found in the Mall dataset's images. Second, some images in the Shanghaitech dataset represented overcrowded spaces (more than 3000 people), which led to instant overfitting when using the MSE loss function during the pre-

Method	MAE
Xception as feature extractor	2.6532
Xception with 15 unfrozen layers	1.8758
Xception with 15 unfrozen layers + data augmentation	1.77
Xception with 15 unfrozen layers + data augmentation + Additional Layers with ReLu activation	2.2872
Xception with 15 unfrozen layers + data augmentation + Additional Layers with Elu activation	1.9273
Xception with pre-train Additional Layers with ELU activation + post-train 15 unfrozen with Additional layers	1.5996

Table 1: Results of regression based approach on Mall Dataset

training procedure. Unfortunately, removing overcrowded images wasn't the best solution due to the small dataset size. To mitigate this problem, we involved two loss functions during pre-training. Initially, we used the MSE loss function, early stopping, and a learning rate scheduler as previously mentioned. Then, we replaced the loss function with MAPE (mean absolute percentage error) and continued the pre-training procedure. Unlike MSE, this loss function is not sensitive to large absolute errors commonly observed in overcrowded images. This method allowed us to significantly improve the scores on the validation part of the Shanghaitech dataset.

After pre-training, we trained the model on the Mall dataset as we had done before. One of the most crucial parts was selecting the number of unfrozen layers both for pre-training and training. This was done by applying the Optuna package, which employs a Bayesian optimization algorithm [1]. The best results are reported in Table 2. The smallest achieved validation MAE was 1.8336. Additionally, we tried to improve this result by adding horizontal flip augmentation. This allowed us to improve the score (MAE: 1.8293). The results are reported in Table 2.

5.2. Density Based Approach

During training, we experimented with different loss functions (euclidean distance and mean squared error), optimizers (SGD and Adam), and learning rates. Additionally, we added early stopping and reduce learning rate on plateau criteria (this one just in combination with SGD). For the learning rates, we experimented with values from 1e-3 to 1e-7. The number of epochs was kept constant at 40. We have decided to freeze all the front-end weights due to having limited GPU resources and trained just the back-end

Method	MAE
Xception with 24 unfrozen layers (pre-training) and 37 unfrozen layers (training)	1.8336
Xception with 15 unfrozen layers (pre-training) and 20 unfrozen layers (training) + Data augmentation	**1.8293**

Table 2: Results of pretraining approach with Shanghai Tech Dataset

layers. The euclidean distance loss didn't perform quite as well as the MSE. It is interesting to note that, even though this function is reported in the CSRNet paper, in their PyTorch implementation, they use MSELoss. The results from the experiments are presented in Table 3. The best configuration was number 3, yielding 4.66 MAE.

Loss	**Optimizer**	**MAE**
Euclidean distance	SGD	22.17
Euclidean distance	Adam	16.63
MSE	SGD	**4.66**
MSE	Adam	5.06

Table 3: Experimental results of applying different loss functions and optimizers on the CSRNet architecture.

By comparing the predicted density maps and checking the worst predictions, it became apparent that the results were unsatisfactory. The training and validation loss curves were decreasing almost perfectly, with no sign of overfitting. However, the limit on the number of epochs (40 compared to 400 in [12]) was preventing the model from learning.

Faced with limited training resources, we resorted to transfer-learning and fine-tuning. For this, we utilized the weights from the CSRNet PyPi [18] package. With straightforward transfer learning, i.e., directly using these weights, we obtained a significant improvement in the test results: MAE = 3.44 and MSE = 17.95. Then, we attempted to finetune these weights by utilizing the training and validation portions of the Mall dataset. While monitoring the train and validation loss during training for 20 epochs, it became apparent from the plots that the model was overfitting soon after the 5th epoch. Therefore, we stopped training and tried inference at that moment. However, the obtained results were unsatisfactory, with an MAE of 6.68.

Finally, we tried to use the whole CSRNet as a feature extractor and train a regressor network by adding a dense layer on top. This resulted in an MAE of **2.72** and MSE of 11.32, which was the best result we obtained using the CSRNet architecture, highlighting the power of dense layers. We attempted to improve this even more by using two Dense layers and Dropout(0.1), but the MAE of this experiment was a little lower (2.75), possibly because the increased number of parameters requires more epochs.

Model	**MAE**
Authors' code	5.18
Transfer Learning	3.44
Fine-tuning	6.68
1 Dense layer Regressor	**2.72**
2 Dense layer + Dropout Regressor	2.75

Table 4: Results of different experiments using the CSRNet architecture: The first row shows the MAE obtained utilizing the official PyTorch implementation of CSRNet (used as a baseline). The second row represents a model with pre-trained weights. The third row is about fine-tuning these weights. The fourth and fifth rows showcase the results of using the CSRNet as a feature extractor and training a network for performing regression.

6. Conclusion

In this study, we investigated two crowd counting approaches: regression-based with the Xception model and density-based using CSRNet. Leveraging pre-trained deep convolutional neural networks, our regression approach achieved exceptional results on the Mall dataset, with the Xception model and strategic fine-tuning yielding the best performance (MAE: 1.5996). Further exploration included pretraining on the Shanghaitech dataset, hinting at potential enhancements in model generalization. Meanwhile, in the density-based approach, CSRNet demonstrated best result (MAE: 2.72) when used as feature extractor to feed training a regressor network.

Future work should consider exploring training and inference times for edge device deployment for real time crowd counting applications, as well as investigating ensemble methods and advanced techniques for improved accuracy and robustness in diverse scenarios.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[2] Ke Chen, Chen Change Loy, Shaogang Gong, and Tony Xiang. Feature mining for localised crowd counting. In *Bmvc*, volume 1, page 3, 2012.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[5] Dejan Dichoski, Suleyman Erim, and Maksim Kokot. Crowd counting github repository. `https://github.com/Di40/CrowdCounting_Xception_CSRNet`, 2024.

[6] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2011.

[7] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[8] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2016.

[10] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2547–2554, 2013.

[11] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. *Advances in neural information processing systems*, 23, 2010.

[12] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes, 2018.

[13] Viet-Quoc Pham, Tatsuo Kozakaya, Osamu Yamaguchi, and Ryuzo Okada. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 3253–3261, 2015.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

[16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015.

[17] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57:137–154, 2004.

[18] Pierre Schaus Xavier Gillard. Csrnet pypi. `https://pypi.org/project/csrnet/#description`, 2023.

[19] 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.