

FUSIONCOUNT: EFFICIENT CROWD COUNTING VIA MULTISCALE FEATURE FUSION

Yiming Ma

Victor Sanchez

Tanaya Guha

Warwick Mathematics Institute
University of Warwick, UK

Department of Computer Science
University of Warwick, UK

School of Computing Science
University of Glasgow, UK

ABSTRACT

State-of-the-art crowd counting models follow an encoder-decoder approach. Images are first processed by the encoder to extract features. Then, to account for perspective distortion, the highest-level feature map is fed to extra components to extract multiscale features, which are the input to the decoder to generate crowd densities. However, in these methods, features extracted at earlier stages during encoding are underutilised, and the multiscale modules can only capture a limited range of receptive fields, albeit with considerable computational cost. This paper proposes a novel crowd counting architecture (*FusionCount*), which exploits the adaptive fusion of a large majority of encoded features instead of relying on additional extraction components to obtain multiscale features. Thus, it can cover a more extensive scope of receptive field sizes and lower the computational cost. We also introduce a new channel reduction block, which can extract saliency information during decoding and further enhance the model’s performance. Experiments on two benchmark databases demonstrate that our model achieves state-of-the-art results with reduced computational complexity.

Index Terms— Crowd density estimation, multiscale feature fusion, efficient crowd counting

1. INTRODUCTION

Crowd counting aims to automatically estimate the number of individuals present in a scene from an image or video. It can be applied in numerous areas, such as traffic control [1], biological studies [2], and recently, social distancing monitoring [3].

Over the years, models for crowd counting have evolved from using classical regression models, such as random forests [4] and Gaussian processes [5], to high-performing convolutional neural networks (CNNs) [6, 7, 8, 9]. These deep nets usually adopt an encoder-decoder approach: First, an image is fed to the encoder to learn data representations (feature maps). The decoder then exploits the highest-level representation (the output from the encoder’s last layer) to generate the density map, which is the distribution of the crowd. Since the convolutional and pooling blocks of VGG networks [10], where each layer exploits kernels of a fixed size, constitute most encoders, the size of receptive fields remains constant across the last encoded feature map. Thus, this representation can only handle images where crowds are of similar scales. However, people are usually depicted in various sizes because of the camera perspective, and as a result, the encoded feature should also have different receptive field sizes to model the scale variation. Multi-column structures [6, 11] have been proposed to solve this problem. However, it has been recently shown that features from each column are almost identical, and training deep models of this type can be very unproductive [7]. Therefore, to solve this scale issue, state-of-the-art methods [8, 12, 13] employ a multiscale module that further processes

the encoded representation and generates a feature map with different receptive field sizes. However, such a strategy ignores that those feature maps extracted by shallower encoding layers already provide information about different scales, and leveraging extra components makes the overall model more computationally expensive.

Hence, our contribution in this paper is a novel multiscale mechanism that addresses the scale issue by leveraging the majority of features generated from the encoder to avoid extra feature-extraction modules and keep the computational cost low. This design incorporates a comprehensive range of receptive field sizes (6 to 192), covering almost all possible scales a person can depict in a crowd image. Experiments on two benchmark databases (ShanghaiTech A & B [6]) demonstrate that our model can achieve state-of-the-art or comparable results with significantly fewer floating-point operations.

2. RELATED WORK

Early crowd counting approaches [14, 15, 16] are based on object detectors, while later works tend to avoid them because of their sensitivity to occlusion and the enormous efforts required to annotate bounding boxes. Some of these non-detection-based methods [4, 5, 17, 18] treat crowd counting as a regression problem: they learn low-level feature representations first, from which the total count is then directly regressed. The training losses of these approaches depend only on the ground-truth count (a scalar) and do not consider crowd density distribution, hence suffering poor generalisation. Thus, these models have soon been superseded by algorithms [19, 20, 8, 12] that instead predict crowd densities, and these density-based methods primarily rely on CNNs.

Since 3D spatial locations have to be projected onto a 2D space while an RGB crowd image is captured from the real world, people can be depicted in different sizes due to perspective distortion, and the variation in scales can severely affect density estimation. To solve this problem, in [21], extra geometric information is exploited to adapt their model to different scenes, but this information is not always provided. Therefore, later methods tend to learn the scales of crowds implicitly. For example, Hydra-CNN [20] divides an image into a pyramid of patches, each representing a different scale and fed to a separate encoder head. Then, all encoded features are concatenated without further processing and utilised to generate the density map. This approach neglects the fact that scales vary continuously across the whole image. CAN [8] is then proposed to address this issue. The whole model involves only one encoder, so a spatial pyramid pooling module [22] is leveraged to make it scale-aware. Then, features are averaged according to learnable weights to ensure that receptive field sizes of the fused feature changes smoothly. However, this architecture is less efficient since it does not exploit low-level features extracted during encoding. These representations,

Encoding

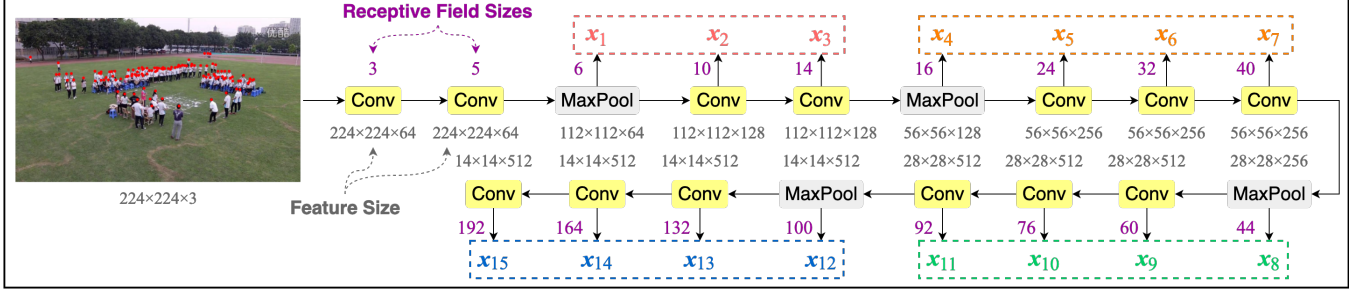


Fig. 1. The encoder of our proposed model FusionCount: Only the first 17 layers of the original VGG-16 are leveraged, and feature maps are collected starting from the third layer. Numbers in purple are features’ receptive field sizes and those tuples ($h \times w \times c$) in gray indicate their sizes, assuming the input image has the size of $224 \times 224 \times 3$. Features with the same spatial resolution are grouped together for the first-phase fusion.

along with the final high-level feature map, can provide information about different scales because they have disparate receptive field sizes. Also, the multiscale module in [8] uses only four filter sizes, thereby covering a limited range of scales.

3. OUR MODEL

This section exhaustively describes our proposed crowd counting model, *FusionCount*. It extensively leverages representations learned during encoding to compute *first-phase* multiscale features, and its decoder further fuses these scale-aware features to generate the density map.

3.1. Encoding

Following common practices [7, 8] in the field, VGG-16 [10] is employed as our model’s encoder. The last max-pooling and fully connected layers are removed since they are accountable for class prediction. Therefore, the encoder comprises 13 convolutional layers and four max-pooling layers. Since the two feature maps extracted before the first max-pooling operation are not sufficiently informative, only features learned afterwards are preserved and fused. Thus, as shown in Fig. 1, the encoder outputs 15 feature maps in total, denoted by \mathbf{x}_j , $j = 1, 2, \dots, 15$. These feature maps are divided into four groups according to their heights and widths: $\mathbf{x}_1 - \mathbf{x}_3$, $\mathbf{x}_4 - \mathbf{x}_7$, $\mathbf{x}_8 - \mathbf{x}_{11}$, and $\mathbf{x}_{12} - \mathbf{x}_{15}$.

3.2. Feature Fusion

Suppose $\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{i+k}$ is a group of feature maps of interest, where \mathbf{x}_i is generated by a max-pooling layer and has c channels, while the rest are produced by convolutional layers and have c^* channels. To assemble a scale-aware representation from them, we adopt a similar strategy as the one proposed in [8], and our method is illustrated in Fig. 2. Given that scales change continuously across an image, the receptive field size of the output multiscale feature, which models the scales, should also be spatially continuous. To accomplish this aim, we fuse these features via weighted averaging, where weight maps are learned from each feature’s spatial importance.

Firstly, since \mathbf{x}_i and \mathbf{x}_{i+j} ($j \in \{1, \dots, k\}$) may not have the same number of channels, we use point-wise convolution to expand \mathbf{x}_i , and denote the output by \mathbf{x}_i^* . Notice that since \mathbf{x}_i is the output of a max-pooling layer and the derivation of \mathbf{x}_i^* does not involve any

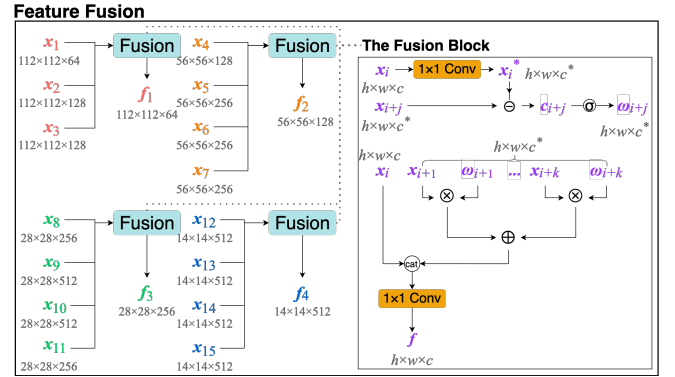


Fig. 2. The feature fusion modules of FusionCount: In each group, weights are computed from contrast features. Then features from convolutional layers are averaged by using these weights and subsequently concatenated with the feature map from the pooling layer.

integration of neighbouring spatial information, \mathbf{x}_i^* is less influenced by background noise. With such an advantage, all other representations are compared against it. Namely, following [8], we propose a similar concept to *contrast features*:

$$\mathbf{c}_j = \mathbf{x}_{i+j} - \mathbf{x}_i^*, \quad (1)$$

with $j = 1, \dots, k$. Since \mathbf{x}_i^* and \mathbf{x}_{i+j} ($j = 1, \dots, k$) have different receptive field sizes, the contrast features \mathbf{c}_j incorporate disparities between any spatial location and its neighbouring pixels. Thus, they can facilitate learning the boundary of each person, thereby determining each receptive field size’s relative importance. We then compute the weights ω_j as follows:

$$\omega_j = \sigma(\mathbf{c}_j), \quad (2)$$

where σ denotes the sigmoid function.¹ Using these weights, $\omega_{i+1}, \dots, \omega_{i+k}$, we then combine $\{\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{i+k}\}$ adaptively and concatenate the averaged feature map with \mathbf{x}_i . Finally, we add a bottleneck layer to reduce computation. This process can

¹Based on experiments (see Section 4.4), the performance of the overall model drops if contrast features are not utilised to compute weights.

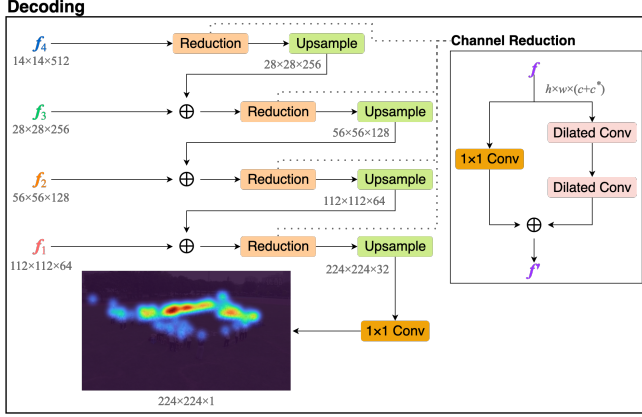


Fig. 3. The decoding process of FusionCount: starting from f_4 , the proposed channel reduction module first decreases its number of channels. The result is then upsampled and fused with another first-phase multiscale feature f_3 .

be expressed as:

$$\mathbf{f} = \text{Conv} \left(\left[\mathbf{x}_i \mid \sum_{j=1}^k \omega_j \odot \mathbf{x}_{i+j} \right] \right), \quad (3)$$

where \mathbf{f} , Conv, “|” and “ \odot ” denote the fused feature, the bottleneck layer, channel-wise concatenation, and element-wise product, respectively.

3.3. Decoding

As shown in Fig. 2, we denote the first-phase multiscale features generated from the four groups via (3) as f_1 , f_2 , f_3 and f_4 . These features are further fused in the reverse order (see Fig. 3). To combine f_4 and f_3 via addition, we first transform the shape of f_4 . Specifically, the number of channels of f_4 is lowered, while its spatial size is expanded. Traditionally, a single point-wise convolutional kernel is used to reduce the number of channels. However, inspired by dilated convolution, which has been shown to have the ability to extract deeper salient information while maintaining the spatial resolution [7], we propose a new channel reduction module. Our two-stream module comprises a dilated convolution block and one bottleneck layer, and the two columns are connected via addition.² To match the spatial resolution of f_3 , we use bilinear interpolation to double the size of f_4 after channel reduction. Then these two features can be fused by summation. The new fused feature is then combined with f_2 following a similar pattern — we first modify its dimensions via the proposed channel reduction module and bilinear interpolation, and then add it to f_2 . This process is followed iteratively until all features are fused. Finally, we feed the final fused feature to the output layer to generate the estimation.

4. EXPERIMENTS

4.1. Datasets

We use the ShanghaiTech A & B datasets [6] for model evaluation and comparison. In ShanghaiTech A, there are 482 crowd im-

²The effectiveness of the combination of the dilated convolution block and the point-wise convolution is verified in Section 4.5.

Table 1. Comparison of our model FusionCount with state-of-the-art models of similar sizes. The best and the second best results are indicated in **bold** and underlined typefaces, respectively.

Model	Mult-Adds	SH A		SH B	
		MAE	RMSE	MAE	RMSE
CSRNet [7]	856.99 G	68.2	115.0	10.6	16.0
CAN [8]	908.05 G	62.3	<u>100.0</u>	7.8	<u>12.2</u>
BL [9]	<u>853.70 G</u>	62.8	101.8	7.7	12.7
DM-Count [23]	<u>853.70 G</u>	59.7	95.7	<u>7.4</u>	11.8
FusionCount (ours)	815.00 G	<u>62.2</u>	101.2	6.9	11.8

ages collected from the Internet. Three hundred of them constitute the training set, and the rest comprise the test set. Scenes in this dataset are highly congested, with an average count of about 501. Also, since images have different resolutions (height and width values range from 182 to 1024), training models on this dataset can be tricky. The size of ShanghaiTech B is larger (716 instances in total; 400 for training and 316 for testing), with an average count of approximately 123. Images from this dataset are taken from a surveillance view in a shopping street and are therefore less crowded. Also, considering that these images have a fixed resolution (768×1024), this dataset is more suitable for real-world applications e.g., public safety monitoring.

4.2. Experiment Settings

During recent years, loss functions based on probability theory, such as Bayes’ theorem and Wasserstein distance, have been shown to help models achieve stronger generalisation capabilities and have thus gained increased popularity. We use the DM-Count loss [23] to supervise the training of FusionCount. An Adam optimiser [24] with a learning rate $1e-5$ and batch size two is leveraged for optimisation. In order to make fair comparisons with other approaches, we use the default data splits. Given that some images have intractably large sizes, from each input image, two patches with a size of 384×512 are cropped and used for training. Our model and its training are implemented in the PyTorch [25] 1.10 framework, and the platform for training is a server with an NVIDIA RTX 3090 GPU and Ubuntu 20.04 LTS OS.

4.3. Results

Following prior works [7, 8, 23], the mean absolute error (MAE) and the root mean squared error (RMSE) of total counts are employed as evaluation metrics. FusionCount is compared with state-of-the-art models that have a similar computational complexity (quantified by the number of multiplications and additions involved in the inference on a 1080×1920 RGB image). In particular, CSRNet [7], CAN [8], BL [9] and DM-Count [23], all of which employ VGGs [10] as encoders, are encompassed for comparison. CSRNet uses dilated convolution in decoding to extract saliency. In addition to this characteristic, CAN also includes a spatial pyramid pooling block to generate multiscale features. BL and DM-Count have no innovations in terms of model architectures, and their contributions are particularly novel loss functions based on probability theory. Table 1 shows detailed performance comparisons. For the case of ShanghaiTech A, partly due to the difficulty in training, FusionCount’s results are not the best but still comparable: it achieves the second-lowest mean absolute counting error. On ShanghaiTech B, FusionCount outper-

forms the other evaluated models under both metrics. These results are remarkable, especially considering our model’s low computational complexity.

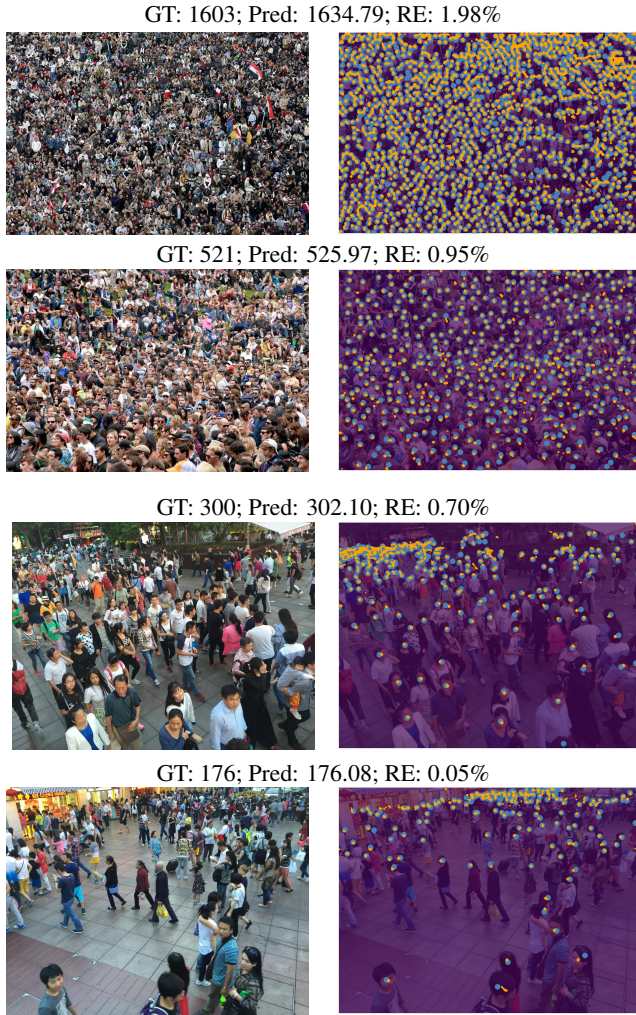


Fig. 4. Qualitative results of FusionCount: Four ground-truth density maps are visualised along with their estimations by our model FusionCount. The top two images are from ShanghaiTech A and the bottom two are from ShanghaiTech B. Ground-truth annotations are marked by blue dots, and the orange shades represent the estimated densities. We also report ‘GT’ i.e., ground-truth counts of people in the images, ‘Pred’ i.e., the predicted counts. ‘RE’ denotes the corresponding relative error.

Four instances from the test sets of ShanghaiTech A & B are depicted in Fig. 4. The left column shows the original input images, and the right column illustrates the ground-truth and the predicted density maps, which are indicated by blue dots and orange shades, respectively. The example in the first row proves that our model can still work well in highly crowded cases. Although in this 600×900 image, there are over 1,600 people, our model still achieves an accurate prediction with a relatively small error (1.98%). Other rows demonstrate that our model is capable of effectively dealing with scale changes. In these images, crowds in the lower parts of the scene have larger scales, while those in the upper part of the scene have

the smallest scales. Our model can make correct predictions for both cases with negligible errors (0.95%, 0.70% and 0.05%, respectively). Thus, all these four instances confirm our model’s strong counting ability.

4.4. Effectiveness of Contrast Features

To confirm that the contrast features can boost the model’s performance, we create a variant for our model, whose only difference from FusionCount is the feature fusion strategy. In this variant, weights are directly generated from encoded features x_{i+j} instead of contrast features c_{i+j} . We train this variant on ShanghaiTech B [6] with the same setting. The MAE and RMSE of this variant are 7.6 and 12.9, respectively, which are larger than those of FusionCount (6.9 and 11.8).

4.5. Ablation Study

This section proves the effectiveness of the channel reduction module proposed in Section 3.3. Experiments are conducted on ShanghaiTech B [6]. The results, which are shown in Table 2, demonstrate that both the point-wise convolutional layer and the two chained dilated convolutional layers are indispensable. Theoretically, they work in a complementary way in reducing the number of channels and extract saliency.

Table 2. Effects of point-wise convolution and the number of dilated convolutional layers in the channel reduction module.

Point-wise Conv	Dilated Conv No.			MSE	RMSE
	2	1	0		
✓	✓			6.9	11.8
✓		✓		7.6	13.0
✓			✓	8.8	15.0
✗	✓			9.5	15.9

5. CONCLUSION

In this paper, we proposed a new crowd counting architecture, FusionCount, which smartly utilises a large majority of features generated during the encoding process to handle perspective distortion. Unlike existing approaches, FusionCount avoids further extraction of multiscale features, thereby significantly reducing overall computation. To this end, We have also improved an existing multiscale fusion mechanism and devised a novel channel reduction block. Experiments on the ShanghaiTech databases demonstrated that FusionCount can outperform relevant state-of-the-art approaches of similar computational complexity. As part of our future work, we are working on accounting for any contextual information in the features fused at the decoding process. Such information can help to more effectively deal with scale changes, as the way the first-stage fusion of encoded features does.

6. REFERENCES

[1] Thomas Moranduzzo and Farid Melgani, “Automatic car counting method for unmanned aerial vehicle images,” *IEEE*

- Transactions on Geoscience and Remote Sensing*, vol. 52, no. 3, pp. 1635–1647, 2014.
- [2] Hao Lu, Zhiguo Cao, Yang Xiao, Bohan Zhuang, and Chunhua Shen, “Tasselnet: counting maize tassels in the wild via local counts regression network,” *Plant methods*, vol. 13, no. 1, pp. 1–17, 2017.
 - [3] Immanuel Jose C. Valencia, Elmer P. Dadios, Alexis M. Fil-lone, John Carlo V. Puno, Renann G. Baldovino, and Robert Kerwin C. Billones, “Vision-based crowd counting and social distancing monitoring using tiny-yolov4 and deepsort,” in *2021 IEEE International Smart Cities Conference (ISC2)*, 2021, pp. 1–7.
 - [4] Victor Lempitsky and Andrew Zisserman, “Learning to count objects in images,” in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds. 2010, vol. 23, Curran Associates, Inc.
 - [5] Antoni B. Chan and Nuno Vasconcelos, “Bayesian poisson regression for crowd counting,” in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 545–551.
 - [6] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma, “Single-image crowd counting via multi-column convolutional neural network,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 589–597.
 - [7] Yuhong Li, Xiaofan Zhang, and Deming Chen, “Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1091–1100.
 - [8] Weizhe Liu, Mathieu Salzmann, and Pascal Fua, “Context-aware crowd counting,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5094–5103.
 - [9] Zhiheng Ma, Xing Wei, Xiaopeng Hong, and Yihong Gong, “Bayesian loss for crowd count estimation with point supervision,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6141–6150.
 - [10] Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015.
 - [11] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su, “Scale aggregation network for accurate and efficient crowd counting,” in *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, Eds., Cham, 2018, pp. 757–773, Springer International Publishing.
 - [12] Pongpisit Thanasutives, Ken-ichi Fukui, Masayuki Numao, and Boonserm Kijisirikul, “Encoder-decoder based convolutional neural networks with multi-scale-aware modules for crowd counting,” in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 2382–2389.
 - [13] Mingjie Wang, Hao Cai, Jun Zhou, and Minglun Gong, “Stochastic multi-scale aggregation network for crowd counting,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 2008–2012.
 - [14] Sheng-Fuu Lin, Jaw-Yeh Chen, and Hung-Xin Chao, “Estimation of number of people in crowded scenes using perspective transformation,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 31, no. 6, pp. 645–654, 2001.
 - [15] Min Li, Zhaoxiang Zhang, Kaiqi Huang, and Tieniu Tan, “Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection,” in *2008 19th International Conference on Pattern Recognition*, 2008, pp. 1–4.
 - [16] Weina Ge and Robert T. Collins, “Marked point processes for crowd counting,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2913–2920.
 - [17] Ke Chen, Chen Change Loy, Shaogang Gong, and Tony Xiang, “Feature mining for localised crowd counting,” in *British Machine Vision Conference, BMVC 2012, Surrey, UK, September 3-7, 2012*, Richard Bowden, John P. Collomosse, and Krystian Mikolajczyk, Eds. 2012, pp. 1–11, BMVA Press.
 - [18] Ke Chen, Shaogang Gong, Tao Xiang, and Chen Change Loy, “Cumulative attribute space for age and crowd density estimation,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2467–2474.
 - [19] Vishwanath A. Sindagi and Vishal M. Patel, “Generating high-quality crowd density maps using contextual pyramid cnns,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1879–1888.
 - [20] Daniel Oñoro-Rubio and Roberto J. López-Sastre, “Towards perspective-free object counting with deep learning,” in *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, Eds., Cham, 2016, pp. 615–629, Springer International Publishing.
 - [21] Di Kang, Debarun Dhar, and Antoni Chan, “Incorporating side information by adaptive convolution,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, Curran Associates, Inc.
 - [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.
 - [23] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai Nguyen, “Distribution matching for crowd counting,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1595–1607, 2020.
 - [24] Diederik P. Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.
 - [25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., pp. 8024–8035. Curran Associates, Inc., 2019.