International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V70I3P208

Volume 70 Issue 3, 66-74, March, 2022

Original Article

Optimizing Crowd Counting on Low Compute Device Using Heterogeneous Convolution Filter

Christopher Alvin[1], Gede Putra Kusuma[2]

[1] *Scholar, Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia.*

[2]*Assistant Professor, Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia.*

[1]christopher.alvin001@binus.ac.id, [2]inegara@binus.edu

Abstract *— The massive growth of population increases potential security risks in the crowded area. To monitor these phenomena, crowd counting can be one of the solutions. But existing crowd counting approach still requires a high compute device. In this work, a mobile device focused density-based crowd counting is proposed. To tackle high resource consumption, the proposed model reduces the number of parameters by using a Heterogeneous Convolution Filter, resulting in lower computation and faster counting time. The experiment is done on three datasets, the ShanghaiTech Part A, ShanghaiTech Part B, UCF_CC_50. Extensive experiments are conducted on mobile devices to have an overview of the performance of the model on mobile devices. The performance of the proposed model compared to CSRNet on mobile are similar on each dataset by only having a bigger MAE by 0.64%, 3%, and 10.56% on UCF_CC_50, ShanghaiTech Part A, and ShanghaiTech Part B, respectively but with better latency by 29.83%, 44.85%, 50%, respectively, and better battery consumption by 25.71%, 44.17% and 50.79% respectively. The proposed model successfully improves the speed of counting the number of people in an image, with a slightly higher MAE compared to the CSRNet on mobile devices.*

Keywords — *Crowd Counting, Convolutional Neural Network, Heterogeneous Convolution Filter, Deep Learning, Mobile Computing.*

I. INTRODUCTION

The rapid growth of the population will increase potential security risks when people gather in the same area [1]. Crowd counting is important for detecting numbers of people in the same area. The purpose of crowd counting is to get a number of people by counting the number of people of a given crowd image or estimating the density of the given image. Because of increasing people density in a region such as large gatherings, regulatory authorities can take precautionary measures, and this can be achieved with crowd counting[2]. There are uptrends death cases caused by a stampede, as shown in Fig. 1 [3].

The problem of crowd counting, or crowd density estimation, is to develop more advanced abilities in analysing crowded scenarios such as crowd monitoring scene understanding. The research to study crowd analysis attracts researchers because of growth in the world population, transmigration, and urbanization contributed to an increased number of crowd activities such as public demonstrations, ceremonies, political rallies [4]. There are many factors affecting the performance of crowd counting. Such as, it is hard to detect people in crowded and dense situations, separate the human body from the background, and the improper distribution of the crowd can decrease the performance [5]. Inconsistent Perspective and Non-Uniform object scale can degrade the performance [6]. Also, illumination and occlusion can degrade crowd counting performance[7].

There are few traditional approaches on crowd counting, Detection-based approaches focused on detecting people in an image that involve a sliding window detector, and the detected people are counted to get the number of the people [4]. Regression-based approaches want to tackle the issues that present in Detection-based approaches that struggle in extremely dense crowds. Regression-based approaches extract features from images and map between the pattern of extracted features and the actual counts [8]. Regression approaches want to address the problems of occlusion and clutter; however, the spatial information is ignored when regressing on the global count. Density estimation-based approaches can accurately estimate the count of people on an image by estimating the image with linear mapping between local patch features and density map that will give the count of objects in that region [9].

Modern approaches to crowd counting are based on CNN methods to count and estimate the density of a picture. CNN-based methods have higher accuracy when handling large density and variation object scales and perspectives. CNN-based methods can be enhanced further by adding scale and contextual information in the model to reduce the error [4].

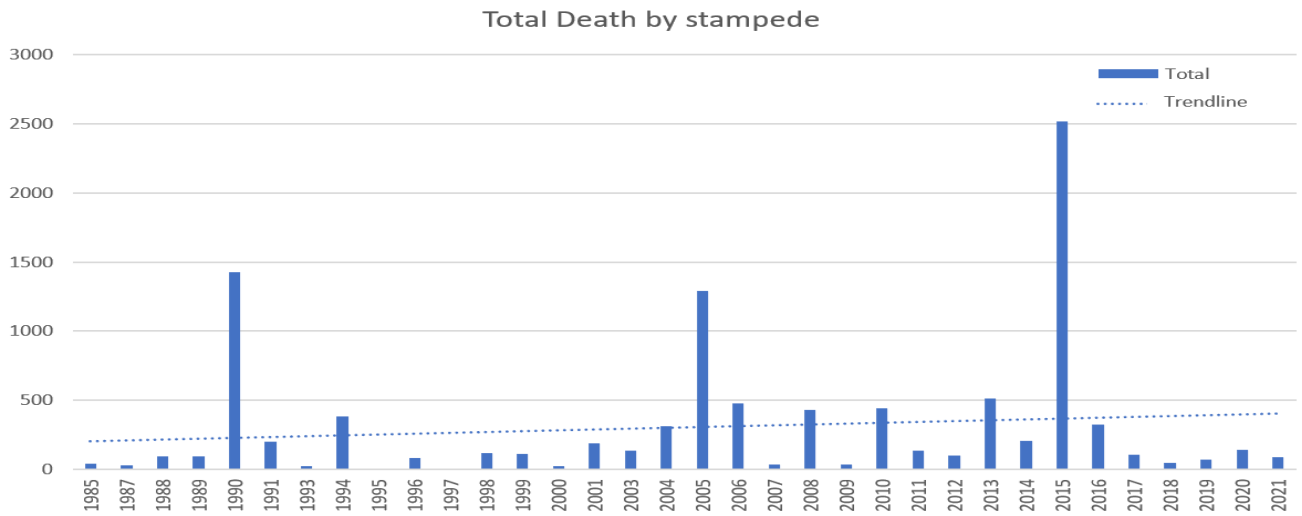

Fig 1. Total Death by stampede

The main challenge is to develop a crowd counting model on a large scale. So, the crowd counting needs to be lightweight and can run on low compute devices to minimize the budget to deploy on a large scale. The model needs to maintain high performance and high accuracy with smaller resources while not compromising on errors. The model needs to run in a lower-compute device to minimize cost and flexibility with the low number of parameters.

The motivation of this research is to develop low compute device crowd detection solutions to detect crowds easily to enhance the security in public places, maintaining the number of people at certain places and times, meanwhile using lower compute devices compared to the state-of-art. This can be achieved by creating a crowd counting model that retains the accuracy of existing model/approaches, with a lower number of parameters, lower latency that determines the speed and lowers the size of the model. The crowd Counting model has been developed using the CNN Density estimation approach with a Heterogenous Convolution Filter. The main contributions of this paper are explained below.

1. Density-estimation crowd counting using Heterogenous Convolution Filter to simplify the model without sacrificing accuracy
2. By simplifying the model, the resource taken to train and run the model should be lowered and make the model can run on lower compute devices.

The rest of the paper is divided into the following sections. Section 2 discuss the existing approach to crowd counting. Section 3 discuss proposed methods. Section 4 discusses experimental analysis and comparison between proposed and existing methods.

II. LITERATURE REVIEW

The current implementation of crowd counting algorithms can be approximately categorized into detection-based methods, regression-based methods, and density estimation-based methods. Below is a detailed explanation of these categories.

A. Crowd Counting

a) Detection-based Methods

In the early development of the crowd counting model, research focused on a detection-based approach. This method involves the use of a sliding window to detect humans in a scene of an image [10]. Usually, this method detects human body parts or full body by classifying features that have been extracted from the human body. This method is struggling in extremely dense crowds because of occlusion and clutter. To overcome this issue, researchers try to process an image to have a feature map at multiple scales by using perspective values to determine scale sizes[11]. There is an approach using multiple scales images and combining the results between the multiple scales to get better results (fusion) [12]. Also, because of the obscured object in a highly dense crowd, a method to detect part of the human body also developed[13].

b) Regression-based Methods

To solve the problem, the previous method that uses a detection-based approach that cannot handle extremely dense crowds and high background clutter, Regression-based approach is being used to map the features extracted from local image patches to count the people inside it (extracted features are directly mapped into a numerical value). One of the advantages of the regression approach is this method avoids the complexity of the detection-based approach that needs to locate of human/individual in an image. This

method captures the low-level features such as edges and foreground pixels then feed them to the regression model but have disadvantages because this approach ignores the spatial information. One of the methods [14] tries to add more features like foreground and textures features given the idea of basic regression-based methods, another researcher [15] also try to add more features by combining SIFT (Scale-invariant feature transform)[16] and Fourier analysis.

c) Density Estimation-based Methods

In Regression, the approach can solve the problem of occlusion and clutter, but the spatial information is ignored. Density estimation-based approaches can accurately estimate the count of people on an image by estimating the image with linear mapping between local patch features and density map that will give the count of objects in that region [9]. This method evolved using CNN

Approach by learning the generated density map of an image. VGG16[17] has been used a few times on this crowd counting problem, for example, CSRNet [18] that use VGG16[17] as a backbone at the first 10 convolutional layers(3x3 kernel) and change the rest of the layers into the dilated convolutional layer to reduce the complexity(3x3 dilated kernel). D-Convnet[19] also utilize VGG16[17] with removal on the 4th max pool layer and addition of 64x1x1 layers on the last layer. The density estimation-based methods have the most focus and have the lowest error among the other methods, but it has been achieved using a deeper network resulting in an increase in counting time and the number of resources used.

B. Heterogeneous Convolution Filter (HetConv)

Heterogeneous Convolution Filter/Layer is made of different sizes of the kernel[20]. A filter can be called heterogeneous when it contains different sizes of the kernel. For example, in a filter that have 256 kernels, some kernel sizes are 3x3 while the rest are 1x1. The other type of kernel, which is Homogenous Kernel, uses using a traditional convolutional filter that can be standard convolution filter, pointwise convolution and depth-wise convolution. A Filter is homogenous when a filter that has 256 kernels with all of the kernels have the same size.

Standard Convolution filter have computational cost at layer L can be given as:

$D_0 \times D_0 \times N$ is the output of the feature map, D_0 is the output square width and height, N is the number of output

$$FL_S = D_0 \times D_0 \times M \times N \times K \times K \quad (1)$$

channels, where M is the number of input channels(depth). For the Number of parts in Figure 2.4, a fraction of 1/P will be $K \times K$, and the remaining $1 - \frac{1}{P}$ total kernels will have the size of 1×1. The total of computational cost of a HetConv layer can be given as:

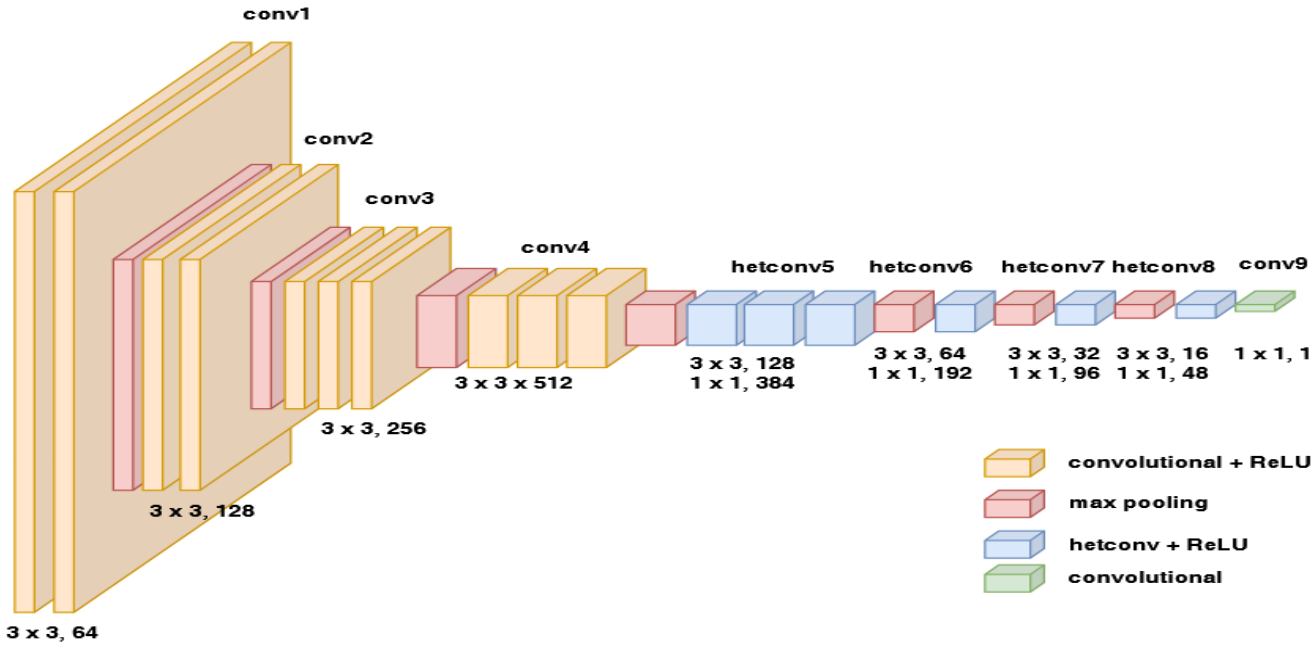

Fig. 2 Proposed CNN Architecture

$$FL_{HetConv} = FL_K + FL_1 \quad (2)$$

Where FL_K is computational cost of $K \times K$ kernels that can be given as:

$$FL_K = \frac{(D_0 \times D_0 \times M \times N \times K \times K)}{P} \quad (3)$$

$FL_1$is computational cost of 1×1 kernels that can be given as:

$$FL_1 = (D_0 \times D_0 \times N) \times (M - \frac{M}{P}) \quad (4)$$

Then we can get the total reduction of computational cost that can be given as:

$$R_{HetConv} = \frac{FL_K + FL_1}{FL_S} = \frac{1}{P} + \frac{(1 - \frac{1}{P})}{K^2} \quad (5)$$

Note that if the value P = 1, it becomes the standard convolutional filter.

The configuration of the filter/kernel in each of the HetConv filters are arranged in a shifted manner. For example, if the first layer starts with a 3x3 kernel in the first position, the second layer will start the 3x3 kernel from the second position and so on. By reducing some kernel on some channels from 3x3 to 1x1, HetConv Layer wants to reduce the spatial extent of a filter, and with 3x3 kernel on some channels, the filter will cover the spatial correlation on some channels, so redundant same spatial correlation can be removed.

III. PROPOSED METHOD

Based on the results of the study in the literature review, it can be concluded that the better the model, usually obtained by adding more layers on the model, thus making it more complex. The idea of the proposed method is to minimize the number of parameters of the CNN model without sacrificing the capability to create high feature density maps. So, the expectation of the model is to have a smaller size, lower number of parameters, lower latency while having a similar MAE.

The flowchart of the proposed method, as shown in Figure 3, consists of four main processes.

1) Load the input dataset
2) Training process through the proposed model
3) The output from the model is the predicted density map that will be used to calculate loss with the generated ground truth

In this section, the proposed architecture will be presented, and then the training process will be presented. The idea of the architecture is from CSRNet, with the dilated convolutional layer is replaced by a Heterogeneous Convolution Layer.

A. CNN Architecture

a) CSRNet

In this research, the main idea comes from CSRNet that replaces 6 layers on VGG16 with a dilated convolutional layer, but in the proposed method, the last 6 layers are replaced with a Heterogeneous Convolution Layer. An additional 1x1 convolutional layer was also added as output.

b) Heterogeneous Convolution Layer (HetConv)

HetConv (Heterogeneous Kernel-Based Convolution) [20], as shown in Figure 2, can reduce the number of parameters as compared to standard convolution operation while maintaining performance. HetConv layer is being used to replace dilated convolution layer to reduce computation and number of parameters and remove spatial information that doesn't need

c) Network Configuration

The architecture that will be used will be the same as in CSRNet [18], but on the first 10 Standard Convolutional layers from VGG-16[17] that CSRNet[18] used, the last 6 layers will be replaced with HetConv layer with P = 4, as illustrated in Figure 2. To compare the difference between density map results from the model and ground truth, Euclidean distance is used. The output of the model is 1/8 size of the input; thus, bilinear interpolation with the factor of 8 is applied, making the output have the same resolution as the input.

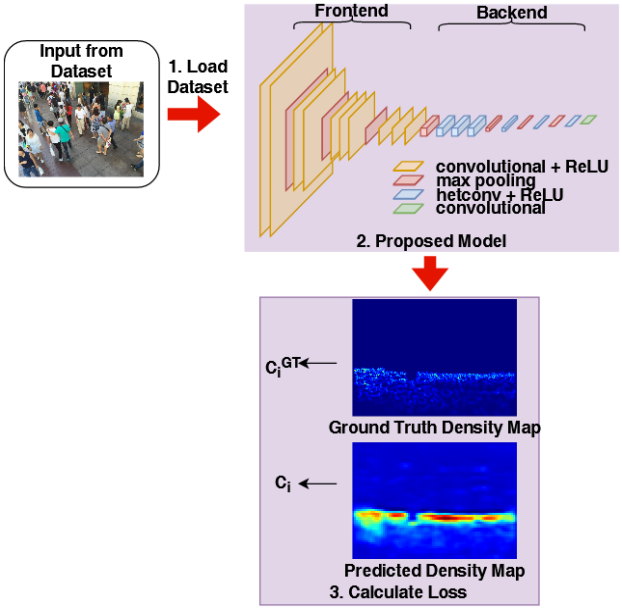

Fig. 3 Flowchart of proposed method

A. Ground Truth Generation

The learning target of the model will be called ground truth. There is info on each image on each dataset that contains the annotation coordinates (x, y) where 1x1 pixel

represents 1 person that is usually marked at the entrance of the head. The information also contains the head size. [21] found the relation between head size and the distance Between the centres of two neighbouring persons in crowded scenes. So, the Geometric adaptive gaussian kernels[21] will be used.

Geometric adaptive gaussian kernels[21] will be used to generate the ground truth by blurring each head with Geometry-Adaptive Gaussian kernel to include the spatial information on the dataset. The setup and parameters on Geometry-Adaptive Gaussian kernel are inherited from CSRNet [18].

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_i) \times G_{\sigma_i}(\boldsymbol{x}),\ with\ \sigma_i = \beta \overline{d_i} \tag{6}$$

B. Data Augmentation

The training process will be divided into 3 parts, preparing the dataset, training, and testing. When preparing the dataset, public datasets such as ShanghaiTech Part A, ShanghaiTech Part B[21], UCF_CC_50[15]. The dataset is already annotated. The next process is generating ground truth using the same configuration as CSRNet [18]. Then augment the dataset by cropping the images into 9 patches at a random location with ¼ size of the image. The first four will contain the four quarters of the image without overlapping.

C. Loss Function

The Lost function that is being used is Mean Squared Error (MSE). Where N is the size of the training batch and Z_i^{Pred} is the output generated by the model with parameters shown. Z_i^{GT} is the ground truth result of the input image.

$$E(\theta) = \frac{1}{2N} \sum_{i=1}^{N} \left\| Z_i^{Pred} - Z_i^{GT} \right\|_2^2 \tag{7}$$

IV. RESULTS AND DISCUSSION

A. Evaluation Metrics

The crowd counting model is evaluated by two commonly used metrics, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

$$MAE = \frac{1}{N} \sum_{i=1}^{N} \left| C_i - C_i^{GT} \right| \tag{8}$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left| C_i - C_i^{GT} \right|^2} \tag{9}$$

N is the number of images, C_i estimated count, and C_i^{GT} is the ground truth count. Performance-related metrics also will be compared, such as number of parameters(million), GFLOPS (Giga Floating Point Operation per Second, a measurement to measure how many floating-point calculations), and model size (MB)

Mobile device related performance such as latency to process an image usually measured in milliseconds (ms), battery usage, CPU Load Peak (%) and Memory Peak (MB) also gathered to measure the real-world performance of the model.

B. Datasets

The crowd counting datasets that are being used in the experiment are ShanghaiTech part A, ShanghaiTech part B and UCF_CC_50[15].

a) ShanghaiTech

Dataset that will be used for this research are ShanghaiTech Part A, ShanghaiTech Part B[21], and UCF_CC_50[15]. ShanghaiTech[21] dataset is divided into two parts; part A is collected from the Internet randomly with a total of 482 images that divided into 300 training images and 182 testing images, with a total of 241,677 labelled people, part B is from street view surveillance camera with a total of 716 images that divided into 400 training images, and 316 testing images, with a total of 88,488 labelled people.

b) UCF_CC_50

UCF_CC_50 [15] is a challenging dataset because of large variation from 94 to 4543 in just 50 images collected from web images that are publicly available; it varies on the contexts and scenes. This dataset is challenging because of the variety of the dataset given such a low number of images. For this dataset, 5-fold cross-validation is performed using the standard approach on [15].

C. Experimental Design

The training and testing are trained using the PyTorch framework using NVIDIA GeForce P100. The SGD optimizer is used on the model with a batch size of 1. the initial learning rate is 1×10^{-7}. The Number of parts of the HetConv layer is 4. For the UCF_CC_50 dataset, 5-fold cross-validation is performed to verify the performance on the model following the standard approach on [15] since the dataset is small. The model will be optimized with a built-in optimized_for_mobile function from PyTorch and will be converted to PyTorch Mobile using the _save_for_lite_interpreter function. After that, the model will be deployed to Samsung Galaxy S8 (SM-G950F). On the mobile, only 1 image IMG_160 from ShanghaiTech Part A dataset will be used to represent the model behaviour on mobile on mobile, and the experiment will focus on resource consumption.

D. Experimental Results

The proposed solution is compared to the state-of-the-art crowd counting approaches. As seen in Table 1, Table 2, and Table 3, overall speed-related performance is compared to the backbone CSRNet; the proposed model has 33.88% fewer parameters, 20,82% GFLOPS, and 33,88% less size because the effect of HetConv Layer that using mixed of 3x3 and 1x1 kernel size. MCNN has the lowest parameters, GFLOPS and model sizes, MAE and RMSE are also compared on Table 1, Table 2, and Table 3, the proposed model is better than MCNN, unfortunately, have higher error compared to more complex CSRNet on all datasets. The significant decrease only results in slightly higher MAE by 2.3 (3.37%), but with lower RMSE by 3.8 (3.30%) over CSRNet in ShanghaiTech Part A, slightly higher MAE by 1.32 (12.45%), and slightly higher RMSE by 2.37 (14.81%) over CSRNet on ShanghaiTech Part B and slightly higher MAE by 4.73 (1.78%), and slightly higher MSE by 25.69 (6.46%) over CSRNet on UCF_CC_50. Density maps produced by the proposed model can be seen in Figure 4.4.

To have an overview of the behaviour of the model in the real use case, the model is deployed to the mobile device. On Table 7, the model performance that converted to the mobile version is tested, resulting in only slight unnoticeable higher MAE and RMSE on every model and dataset, but has a smaller size compared to the full desktop model, by 49.89% on the proposed model, 49.97% on CSRNet and 7.21% on MCNN. The test is conducted using 5 images taken from each dataset 5 times on every compared model.

Table 1. Result on ShanghaiTech Part A compared with other models on PC

Dataset	Model	MAE	RMSE	Number of Parameters	GFLOPS	Size
ShanghaiTech Part A[21]	MCNN [21]	110.2	173.2	0.13 M	56.21	0.596 MB
	CSRNet[18]	68.2	115.0	16.26 M	857.84	130.13 MB
	Proposed Model	**70.5**	**111.2**	**10.75 M**	**679.2**	**86.03 MB**

Table 2. Result on ShanghaiTech Part B compared with other models on PC

Dataset	Model	MAE	RMSE	Number of Parameters	GFLOPS	Size
ShanghaiTech Part B[21]	MCNN [21]	26.4	41.4	0.13 M	56.21	0.596 MB
	CSRNet[18]	10.6	16.0	16.26 M	857.84	130.13 MB
	Proposed Model	**11.92**	**18.37**	**10.75 M**	**679.2**	**86.03 MB**

Table 3. Result on UCF_CC_50 compared with other models on PC

Dataset	Model	MAE	RMSE	Number of Parameters	GFLOPS	Size
UCF_CC_50[15]	MCNN [21]	377.6	509.1	0.13 M	56.21	0.596 MB
	CSRNet[18]	266.1	397.5	16.26 M	857.84	130.13 MB
	Proposed Model	**270.825**	**423.189**	**10.75 M**	**679.2**	**86.03 MB**

Table 4. Resource Consumption on Mobile Devices on ShanghaiTech Part A

Dataset	Model	Average CPU Peak (%)	Average Memory Peak (MB)	Average Battery (mAH)	Average Latency (s)
ShanghaiTech Part A[21]	MCNN[21]	45.04 %	821.54 MB	1.45 mAH	3.85 s
	CSRNet[18]	51.12 %	722.50 MB	3.69 mAH	12.53 s
	Proposed Model	**50.36 %**	**604.61 MB**	**2.06 mAH**	**6.91 s**

Table 5. Resource Consumption on Mobile Device on ShanghaiTech Part B

Dataset	Model	Average CPU Peak (%)	Average Memory Peak (MB)	Average Battery (mAH)	Average Latency (s)
ShanghaiTech Part B[21]	MCNN[21]	46.32 %	1320 MB	2.47 mAH	10.2 s
	CSRNet[18]	51.88 %	1036 MB	7.62 mAH	34.66 s
	Proposed Model	**50.04 %**	**1004 MB**	**3.75 mAH**	17.33 s

Table 6. Resource Consumption on Mobile Device on UCF_CC_50

Dataset	Model	Average CPU Peak (%)	Average Memory Peak (MB)	Average Battery (mAH)	Average Latency (s)
UCF_CC_50[15]	MCNN[21]	46.48 %	1146.54 MB	2.32 mAH	5.96 s
	CSRNet[18]	50.68 %	834.65	4.24 mAH	18.77 s
	Proposed Model	50.96 %	842.08 MB	3.15 mAH	13.17 s

Table 7. Result on Model converted to Run on Mobile Device

Dataset	Model	MAE	RMSE	Size
ShanghaiTech Part A[21]	MCNN[21]	110.81	174.15	0.553 MB
ShanghaiTech Part A[21]	CSRNet[18]	68.57	116.16	65.1 MB
ShanghaiTech Part A[21]	**Proposed Model**	**70.69**	**112.1**	**43.1 MB**
ShanghaiTech Part B[21]	MCNN[21]	26.77	41.98	0.553 MB
ShanghaiTech Part B[21]	CSRNet[18]	10.75	16.49	65.1 MB
ShanghaiTech Part B[21]	**Proposed Model**	**12.02**	**18.66**	**43.1 MB**
UCF_CC_50[15]	MCNN[21]	382.87	516.2	0.553 MB
UCF_CC_50[15]	CSRNet[18]	269.81	409.69	65.1 MB
UCF_CC_50[15]	**Proposed Model**	**271.54**	**426.62**	**43.1 MB**

In Table 4, the proposed model has been tested on ShanghaiTech Part A dataset. The proposed model has a lower average CPU peak, average memory peak, average battery consumption, and average latency by 0.76%, 117.89 MB, 8.18 mAH and 34.56s compared to CSRNet, respectively. Compared to MCNN, the proposed model has a higher average CPU peak of 5.32%, a lower average memory peak of 216.93MB, higher average battery consumption by 3.02 mAH and higher average latency of 15.31s.

In Table 5, the proposed model is tested on the ShanghaiTech Part B dataset. The proposed model has a lower average CPU peak, average memory peak, average battery consumption, average latency by 1.84%, 32 MB, 19.34 mAH and 86.68s compared to CSRNet, respectively. Compared to MCNN, the proposed model has a higher average CPU peak of 3.72%, a lower average memory peak of 316 MB, higher average battery consumption by 6.4 mAH and higher average latency of 35.58s.

In Table 6, the proposed model is tested on the UCF_CC_50 dataset. The proposed model has lower average battery consumption by 5.42 mAH, lower average latency by 28s, higher average CPU peak by 0.28, and higher memory peak by 7.43 MB compared to CSRNet. Compared to MCNN, the proposed model has a higher average CPU peak of 4.48%, a lower average memory peak of 304.46 MB, higher average battery consumption by 4.14 mAH and higher average latency by 36.02s.

Figure 4 shows an example density map predicted by MCNN, CSRNet, and the proposed method. The density maps generated by MCNN are the most blurred among the others, and the estimated crowd count is the most inaccurate. On the contrary, the density maps generated by CSRNet are the most detailed, and the proposed method is blurrier than the CSRNet due to the usage of the HetConv layer, but the estimated crowd count on CSRNet are similar to the proposed method.

E. Trade-off Analysis Between Speed and Accuracy

To obtain more speed, fewer parameters and GFLOPS is required since it measures computation operations on the device where the model will be deployed. The proposed model can outperform MCNN[21] in terms of accuracy but have speed limits. MCNN[21] have better speed because of the configuration of the model that consists of a simple conv-pooling-conv-pooling multi-column structure. On the other

hand, the proposed model outperforms CSRNet in terms of speed, but there is some reduction of performance given the baseline model of CSRNet. The speed advantages are because the HetConv kernel is removing more unnecessary information compared to Dilated Convolution Kernel used in CSRNet. Table 4,5,6 shows that the CPU peak and memory peak on every dataset on proposed model, CSRNet, and MCNN have similar pattern, the memory peak influenced a lot by higher resolution on given image, meanwhile the battery consumption is influenced by the latency to process an image. The latency represents the complexity of the model.

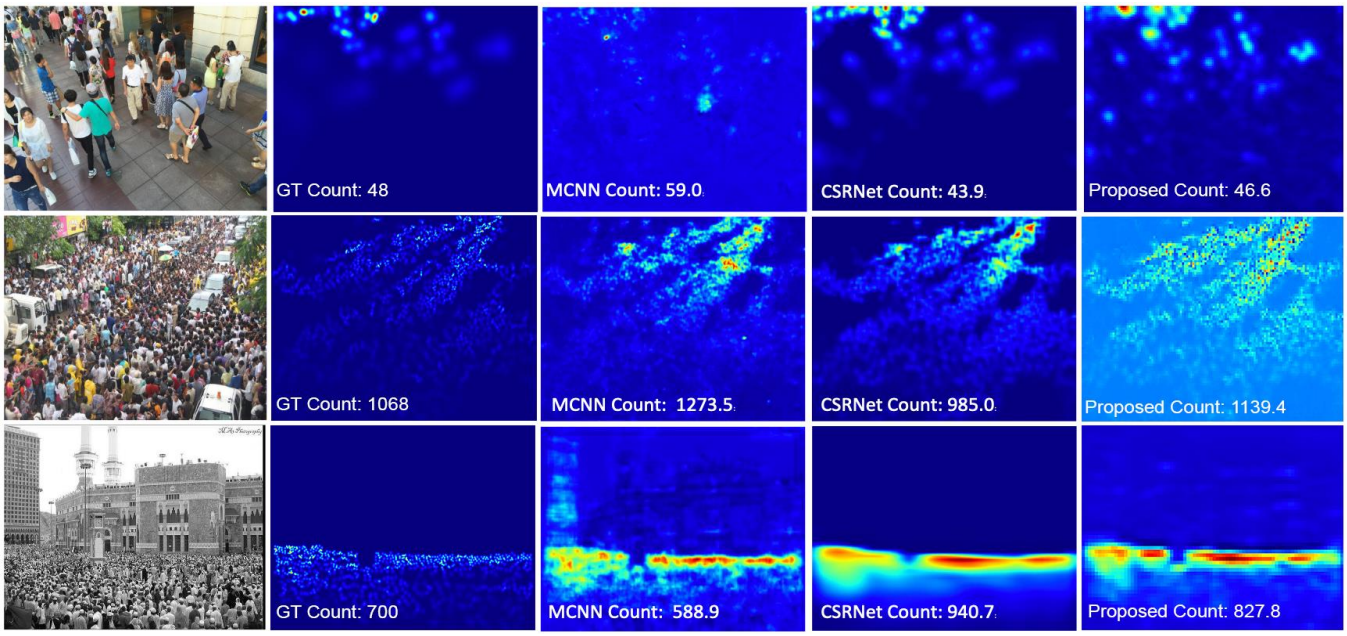

Fig. 4 Example experimental results

V. CONCLUSION

In this work, crowd counting models that utilize Heterogeneous Convolution filters are proposed to focus on improving the crowd counting model on mobile devices. The main results and findings are concluded as follows.

(1) The proposed model successfully reduces a significant amount of computation and improve the significant speed of counting the number of people in an image compared to the CSRNet on mobile devices.
(2) The proposed model has a slightly higher error compared to CSRNet, but since the speed improvement is big, the higher error can be compromises
(3) The uses of HetConv (Heterogeneous Kernel-Based Convolution) Kernel can reduce computation, improving the speed of counting, with slightly higher error
(4) Less complex (lightweight) networks have higher errors compared to more complex network

Although the proposed methods have been achieved to lower the complexity of the model, they can be further improved in future work. The current existing state of the art are usually using complicated CNN architecture while achieving good MAE; the resource consumption and latency are high. A lightweight network can be the solution to achieve lower resource consumption and latency. However, designing a lightweight network usually results in MAE/accuracy drop. Designing a lightweight, the efficient network is needed to reduce resource consumption and latency without sacrificing MAE/accuracy drop

REFERENCES

[1] X. Ma, S. Du, and Y. Liu, A Lightweight Neural Network For Crowd Analysis Of Images With Congested Scenes, in 2019 IEEE International Conference on Image Processing (ICIP), (2019) 979–983, doi: 10.1109/ICIP.2019.8803062.
[2] A. Unnikrishnan, A. Narayan, and H. Guruprasad, A Review of the Advances in Population Estimation and Human Behaviour Estimation Using Computer Vision and Mobile Phone Data, Int. J. Comput. Trends Technol., 46(1) (2017) 37–41. doi: 10.14445/22312803/IJCTT-V46P108.
[3] List of human stampedes and crushes, Wikipedia. https://en.wikipedia.org/wiki/List_of_human_stampedes_and_crushes (accessed Oct. 08, 2021).
[4] V. A. Sindagi and V. M. Patel, A survey of recent advances in CNN-based single image crowd counting and density estimation, Pattern Recognit. Lett., 107 (2018) 3–16. doi: 10.1016/j.patrec.2017.07.007.
[5] A. Khan, J. Ali Shah, K. Kadir, W. Albattah, and F. Khan, Crowd Monitoring and Localization Using Deep Convolutional Neural Network: A Review, Appl. Sci., 10(14) (2020). 4781. doi: 10.3390/app10144781.
[6] N. Ilyas, A. Shahzad, and K. Kim, Convolutional-Neural Network-Based Image Crowd Counting: Review, Categorization, Analysis, and Performance Evaluation, Sensors, 20(1) (2019) 43. doi: 10.3390/s20010043.

[7] H. H. Lin and K. T. Win, HPC Feature for Crowd Outdoor Scenes Estimation, Int. J. Comput. Trends Technol.,67(8) (2019) 1–6., doi: 10.14445/22312803/IJCTT-V67I8P101.

[8] C. C. Loy, K. Chen, S. Gong, and T. Xiang, Crowd Counting and Profiling: Methodology and Evaluation, (2013) 347–382.

[9] V. Lempitsky and A. Zisserman, Learning To Count Objects in Images, in Advances in Neural Information Processing Systems, 23 (2013) [Online]. Available: https://proceedings.neurips.cc/paper/2010/file/fe73f687e5bc5280214e0486b273a5f9-Paper.pdf.

[10] S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Trans. Pattern Anal. Mach. Intell., 39(6) (2017)1137–1149. doi: 10.1109/TPAMI.2016.2577031.

[11] S. Basalamah, S. D. Khan, and H. Ullah, Scale Driven Convolutional Neural Network Model for People Counting and Localization in Crowd Scenes, IEEE Access, 7 (2019) 71576–71584.doi: 10.1109/ACCESS.2019.2918650.

[12] D. Babu Sam, S. V. Peri, M. Narayanan Sundararaman, A. Kamath, and V. B. Radhakrishnan, Locate, Size and Count: Accurately Resolving People in Dense Crowds via Detection, IEEE Trans. Pattern Anal. Mach. Intell., (2020) 1–1. doi: 10.1109/TPAMI.2020.2974830.

[13] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, Object Detection with Discriminatively Trained Part-Based Models, IEEE Trans. Pattern Anal. Mach. Intell.,32(9) (2010) 1627–1645.Sep. 2010, doi: 10.1109/TPAMI.2009.167.

[14] A. B. Chan and N. Vasconcelos, Bayesian Poisson regression for crowd counting, in 2009 IEEE 12th International Conference on Computer Vision, (2009) 545–551. doi: 10.1109/ICCV.2009.5459191.

[15] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, Multi-source Multi-scale Counting in Extremely Dense Crowd Images, in 2013 IEEE Conference on Computer Vision and Pattern Recognition, (2013) 2547–2554, doi: 10.1109/CVPR.2013.329.

[16] D. G. Lowe, Object recognition from local scale-invariant features, in Proceedings of the Seventh IEEE International Conference on Computer Vision, 2 (1999) 1150–1157 doi: 10.1109/ICCV.1999.790410.

[17] K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, (2014). [Online]. Available: http://arxiv.org/abs/1409.1556.

[18] Y. Li, X. Zhang, and D. Chen, CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes, in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, (2018) 1091–1100, doi: 10.1109/CVPR.2018.00120.

[19] Z. Shi et al., Crowd Counting with Deep Negative Correlation Learning, in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, (2018) 5382–5390, doi: 10.1109/CVPR.2018.00564.

[20] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri, HetConv: Heterogeneous Kernel-Based Convolutions for Deep CNNs, in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (2019) 4830–4839, doi: 10.1109/CVPR.2019.00497.

[21] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, Single-Image Crowd Counting via Multi-Column Convolutional Neural Network, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016). 589–597. doi: 10.1109/CVPR.2016.70.