Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# *LC-Net*: *L*ocalized *C*ounting *Net*work for Extremely Dense Crowds

Tarik Reza Toha [a,*,1], Najla Abdulrahman Al-Nabhan [b,1], Saiful Islam Salim [a], Masfiqur Rahaman [a], Uday Kamal [c], A.B.M. Alim Al Islam [a]

[a] *Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh*
[b] *Department of Computer Science, King Saud University, Riyad, Saudi Arabia*
[c] *Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh*

ARTICLE INFO

*Article history:*
Received 14 May 2021
Received in revised form 24 March 2022
Accepted 20 April 2022
Available online 4 May 2022

*Keywords:*
Crowd localization
Deep learning
Computer vision
Hajj pilgrimage
Public safety and security
Crowd monitoring
Real implementation

ABSTRACT

Large mass gatherings such as pilgrimages, protests, etc., often pose serious challenges for the crowd management personnel to maintain public safety and security especially in dense crowds. These challenges can be mitigated through estimating the number of attendees as well as localizing them in a particular crowded event, where existing research studies are yet to provide accurate information in an efficient manner. Therefore, in this paper, we propose a novel deep learning architecture namely LC-Net to precisely and efficiently locate as well as count the attendees in dense crowds using a crowd localization map. Here, we exploit the notions of residual layers and dilated convolution to improve both the accuracy and efficiency of our architecture. Besides, we propose a new data augmentation technique to resize the high-resolution training images based on crowd density that substantially boosts our localization accuracy. Rigorous experimental evaluation of our proposed LC-Net over four different public crowd datasets such as NWPU-Crowd, UCF-QNRF, ShanghaiTech-A, and ShanghaiTech-B shows a substantial performance improvement while using LC-Net in terms of precision and recall in most of the cases. The improvement eventually results in an improved F1 score in all cases compared to the state-of-the-art approaches. Further, we present a real implementation of our proposed approach using a client–server application. In the server, we execute the LC-Net model over the images captured in real-time using an IP Camera and then visualize the results in a graphical manner. This implementation demonstrates the applicability of our proposed approach in real cases.

## 1. Introduction

A myriad number of crowded events take place every year around the world for different purposes such as religious events, sports events, political events, cultural events, etc. Among these events, Hajj, the Islamic pilgrimage to Mecca, Saudi Arabia, is perhaps the largest and most long-standing annual mass gathering event on the earth [1]. According to General Authority for Statistics, approximately 2.5 million pilgrims attended this five-day long pilgrimage in 2019 [2]. Mismanagement in such large mass gatherings can lead to catastrophic results such as stampede [3]. In order to ensure public safety and security in such large mass gatherings, estimating the number of people along with their locations can facilitate the crowd monitoring activities performed by the management personnel [4].

To estimate the number of people in a crowded area, most of the existing research studies estimate a person-density map followed by computing integral over this map to yield a final count [5,6]. Although this approach results in a precise count of people in a crowd, it cannot provide location information of a person in the crowd [7]. Such location information is required for performing crowd monitoring activities such as pedestrian tracking [4], crowd flow estimation [8], etc. There exists a few research work [4,7] found in the literature that provide location information of a person in a highly crowded event. However, due to their architectural limitations, these approaches exhibit poor localization accuracy for initiating a crowd tracking algorithm [9]. Besides, these approaches demand higher computational resources that makes them inappropriate in different cases such as real-time deployment [10].

As a remedy of the aforementioned problems, we propose a novel deep learning architecture named Localized Counting Network (LC-Net) that uses the crowd localization map to locate a person precisely in dense crowds. Here, we exploit residual layers and dilated convolution to improve both the accuracy and efficiency of our architecture. In addition, we propose a

* Corresponding author.
*E-mail addresses:* 1017052013@grad.cse.buet.ac.bd (T.R. Toha), nalnabhan@ksu.edu.sa (N.A. Al-Nabhan), 1018052067@grad.cse.buet.ac.bd (S.I. Salim), 0421052008@grad.cse.buet.ac.bd (M. Rahaman), udday2014@gmail.com (U. Kamal), alim_razi@cse.buet.ac.bd (A.B.M.A.A. Islam).
[1] Both authors contributed equally to this research.

https://doi.org/10.1016/j.asoc.2022.108930
1568-4946/© 2022 Elsevier B.V. All rights reserved.

novel density-based image resizing technique for data augmentation used in the training stage that substantially boosts the localization accuracy. We evaluate the performance of LC-Net against state-of-the-art crowd localization approaches over four different public crowd datasets namely NWPU-Crowd [8], UCF-QNRF [4], ShanghaiTech-A [6], and ShanghaiTech-B [6]. Our evaluation demonstrates a substantial performance improvement in terms of precision and recall in most of the cases, which eventually results in an improved F1 score in all the cases. Based on our work, we make the following set of contributions in this paper:

- We propose a novel end-to-end trainable deep learning architecture named LC-Net for accurate crowd localization in an efficient manner through exploiting residual layers and dilated convolution.
- We propose a new data augmentation technique to resize the high-resolution images based on crowd density used in the model training stage.
- We compare the performance of LC-Net against the state-of-the-art crowd localization approaches over two most recent and comprehensive crowd datasets namely NWPU-Crowd and UCF-QNRF, and achieve substantial improvement in F1 score. Besides, we compare our proposed LC-Net over two older and less comprehensive datasets namely ShanghaiTech-A and ShanghaiTech-B to confirm the robustness of our proposed method.
- For real-world implementation, we propose a client–server application, where a Raspberry Pi (client) captures the crowd images and uploads them to a web server. In the server, we perform crowd localization tasks using LC-Net. Finally, we visualize the crowd counting values as per the proposed client–server to demonstrate the applicability of our proposed approach.

We organize the rest of this paper in the following way. In Section 2, we will discuss the research studies related to this paper. After that, we will show the proposed approach and discuss the technical details of LC-Net architecture in Section 3. Next, we will discuss the corresponding training methodology in Section 4. We show the experimental results in Section 5. Next, we will present a real implementation of our proposed approach in Section 6. Finally, we will conclude this paper mentioning some future work.

## 2. Related work

In earlier stages of crowd analysis, various basic machine learning and computer vision algorithms such as detection, regression, and density-based approaches were developed to predict crowd density maps [11]. However, these methods cannot handle various challenges such as variations in scale and perspective, occlusions, non-uniform density, etc. Over the last few years, with the help of convolutional neural network (CNN), researchers mitigate these challenges over a wide range of scenarios.

In case of a crowd counting technique adopting a density map-based approach, the ground truth density map is generated by defining a normalized Gaussian distribution around each annotated person [7]. Through learning this density map using convolutional neural network, state-of-the-art methods can predict the crowd count with lower error over public crowd datasets. For example, Li et al. [5] proposed an efficient crowd counting network named Congested Scene Recognition Network (CSRNet) through adding a Dilation module on top of the VGG-16 backbone. Besides, Liu et al. [12] proposed a context-aware crowd counting network through combining the features of multiple streams using different respective field sizes. On the other hand, Xiong et al. [13] proposed an efficient crowd counting network named Spatial Divide-and-Conquer Network (S-DCNet) that transforms open-set counting into a closed-set problem via dividing a dense image until the crowd count of sub-images becomes similar to previously observed closed set. Nonetheless, Ma et al. [14] proposed a novel Bayesian loss function for crowd counting that constructs a density contribution probability model from the point annotations.

The common limitation of the aforementioned approaches is that they cannot provide reliable location information of the people in the crowd that is required for crowd monitoring activities such as crowd tracking and crowd flow estimation [8]. To address this issue, Idrees et al. [4] proposed a novel composition loss function for counting, density map estimation, and localization in dense crowds. In their architecture, they used DenseNet-201 [15] as the backbone that requires higher computational resources [16] making them inappropriate in different cases such as real-time deployment [17]. On the other hand, Liu et al. [7] proposed a novel crowd counting and localization framework named Recurrent Attentive Zooming Network (RAZ_Net) that recurrently detects ambiguous image region and zooms it into a high resolution for re-inspection. In their architecture, they used VGG-16 [18] as the backbone, which suffers from vanishing gradient problem [19] that causes poor localization accuracy resulting in high false outputs. As a remedy of the aforementioned inefficiency and low accuracy problems, we propose a novel deep learning architecture named LC-Net in this paper.

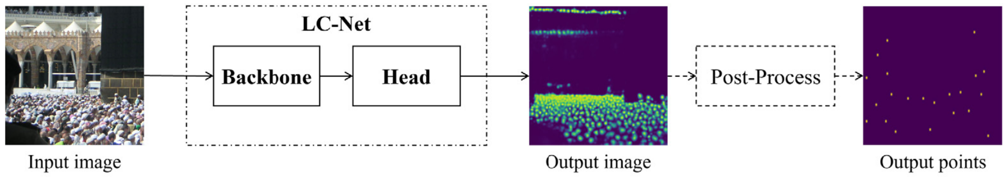
## 3. Our proposed architecture

Our proposed Localized Counting Network (LC-Net) has two parts namely backbone network and head network. The backbone network extracts low-level information from raw images and forwards them to the head network. The head network learns high-level information and localizes people from the crowd images. Fig. 1(a) shows the general structure diagram and Fig. 1(b) shows the detailed network architecture of LC-Net with their inter-connectivity. Next, we describe both the backbone and head networks in detail.

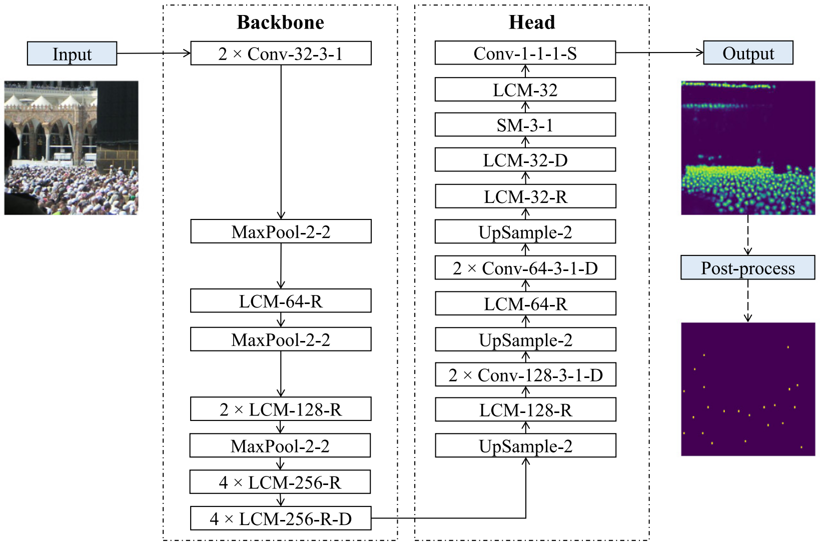
### 3.1. Backbone network

In order to perform accurate and efficient crowd localization, we design a novel highly precise convolutional structure named as Localized Counting Module (LCM) and use several instances of the designed LCM modules in our backbone network. In an LCM module, we learn some features from the input image. After that, we summarize the features to reduce irrelevant features, and then aggregate them with original features. In this process, we localize the persons accurately in an efficient manner. Fig. 2(a) shows the block diagram of an LCM module. We can see that one LCM module contains one convolutional module and $N$ number of residual blocks (also known as bottleneck). In a residual block, one point convolution layer having $1 \times 1$ kernel size is used before a $3 \times 3$ convolution layer to compress the feature representation that enhances the learning ability of the network [20]. After that, one shortcut connection is used to aggregate the compressed output with the previous layer's output. These shortcut connections alleviate the vanishing gradient problem at deeper layers [19] that results in better accuracy. In Fig. 2, LCM-32-R denotes an LCM module having residual layers and LCM-32 denotes an LCM module having no residual layer.

In our backbone network, we use mostly $3 \times 3$ filters in the convolutional modules and double the number of channels after every pooling step. Fig. 2(c) shows the structure of a convolutional module used in LC-Net. Here, Conv-32-3-1 denotes a convolution layer having 32 filters, $3 \times 3$ kernel size, and one



(a) General structure diagram of LC-Net



(b) Detailed network architecture of LC-Net

**Fig. 1.** Block diagram of our proposed LC-Net architecture.

stride. After each convolutional layer, we use batch normalization in order to stabilize training, speed up convergence, and regularize the model [21]. Moreover, we use Leaky Rectified Linear Unit (LeakyReLU) layer as the activation function to avoid the dying problem caused by classical ReLU [22].

We provide 512 × 512 RGB image patches to the input layer and get 64 × 64 feature maps from the backbone. In order to downsample the image patch, we use MaxPooling operation instead of strided convolution to reduce the learning parameters. We use only three MaxPooling operations to abstract our images such that the output feature map of our feature extractor contains enough spatial information for localization. In deeper layers, we use dilated convolution to expand the receptive field without losing resolution as suggested in [5]. We denote such dilated LCM blocks in Fig. 1(b) using the postfix 'D' such as LCM-256-R-D.

### 3.2. Head network

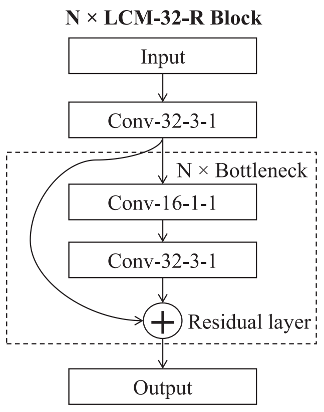
The head network receives feature map from the backbone network as its input and generates a final prediction as its output. In the head network, we apply dilation operation to learn deep features without decreasing the resolution. Thus, we increase the accuracy of our network in an efficient manner. We present the summarized architecture of the head network in Fig. 1(b). Here, we use three UpSampling layers to increase the resolution of the feature map and we get the same input resolution as output. Note that, we do not use strided transpose convolution to upsample the feature map in order to reduce the network complexity. Besides, after each UpSampling layer, we halve the number of channels and provide one LCM block and one pair of dilated convolutional modules to expand the receptive field. After the third UpSampling layer, the network learns rich information similar to the ground truth feature map during the training stage. Hence, we provide some additional layers to learn more high-level information from the feature map. For example, we use two LCM modules having no residual layer (Fig. 2(b)) and one sharpening module 2(d). The sharpening module (SM) contains one AveragePooling layer followed by one MaxPooling layer, where both of them have 3 × 3 pool size and one stride. The former layer enlarges the value of true location and suppresses the noises of a feature map. On the other hand, the latter layer highlights the strongest features and suppresses the weakest ones. Thus, this module sharpens the internal feature map, which helps the LC-Net to achieve good precision. Finally, we add one point convolutional layer with sigmoid activation function for binary classification. It is denoted in Fig. 1(b) using the postfix 'S'. Next, we describe our training method of this whole deep learning architecture.

### 3.3. Specialty of our proposed architecture

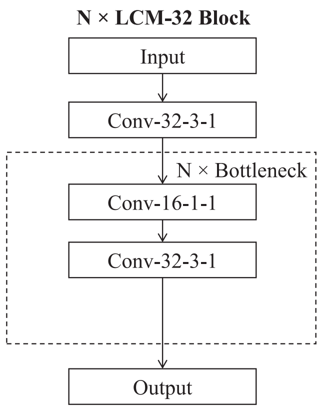
The specialty of our proposed LC-Net architecture is that we jointly use residual layers and dilated convolution. The residual layers reduce the computational cost of the architecture that improves the efficiency of the LC-Net without lowering the accuracy. On the other hand, the dilated convolution learns the deep features without losing the resolution of the feature maps. It enhances both the accuracy as well as the computational efficiency of LC-Net. By using these special architectural blocks, our LC-Net can locate the persons precisely in an efficient manner.

## 4. Training method

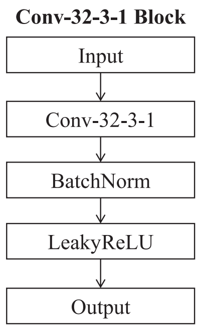
During the training stage, LC-Net learns the mapping between input image patches and ground truth feature maps. Prior to



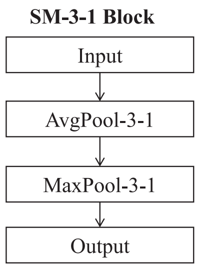
(a) Localized Counting Module with Residual layer



(b) Localized Counting Module without Residual layer



(c) Convolutional Module



(d) Sharpening Module

**Fig. 2.** Block diagram of different modules used in LC-Net.

training, we need to pre-process our training dataset and corresponding ground truth. We describe the pre-processing and training stages in detail in the next subsections.

### 4.1. Data pre-processing

In the pre-processing stage, we prepare the input image patch and corresponding ground truth to feed the LC-Net. We divide our input image into a grid of 512 × 512 pixels. Here, we use zero-padding if the dimensions of an image are not divisible by 512. On the other hand, we use plus (+) annotated feature map as the ground truth of a particular image patch, since the plus annotation achieves better localization accuracy than dot annotation as stated in [7]. In a dot annotated feature map, one dot per person is used. Around each dot, the plus annotation adds a small neighborhood that helps the model to learn better. Although the existing dataset provides only dot annotation, we can convert them into plus annotation through a simple 2D convolution provided by OpenCV library [23]. Here, we need to use the following 3 × 3 kernel, $K = [[0, 1, 0], [1, 1, 1], [0, 1, 0]]$. We perform this patch generation process only once, and save both the image patch and corresponding feature map patch for further use. In addition to patch generation, we calculate the channel-wise mean and standard deviation of the image pixel values of the dataset for data augmentation purpose. Next, we describe the training stage of LC-Net along with the implementation details.

### 4.2. Training details

As our proposed LC-Net is an end-to-end trainable structure, we adopt a very straightforward way to train the LC-Net over the training dataset. Prior to training, we perform some data augmentation tasks to increase the model performance. For example, we standardize the image data to zero mean and unit variance using our previously calculated mean and standard deviation of pixel values. After that, we perform horizontal flip operation using a uniform random distribution. If we get a head for an image, we flip both the image and the corresponding feature map. Next, we construct our proposed CNN architecture LC-Net using Keras library [24]. We set the initial random weights of all the Keras layers using He-Uniform distribution [25], since this initializer helps a ReLU activated network to perform better. Here, we provide a seed for this distribution to get reproducible results.

We represent a person in our ground truth feature map using a plus shape (five ones). If there is no person, the corresponding pixels are zero in the ground truth map. Hence, our ground truth feature map is a binary 2D array. In order to learn this binary map, we use binary cross-entropy as the loss function. Here, the number of positive classes is much lower than the number of negative classes. To compensate this class imbalance, we set the weight of the positive class to 100 as adopted in [7]. On the other hand, we use a custom metric named 'localized counting error' to evaluate our model during training time. Here, we calculate the absolute counting error between ground truth and predicted map after localization. For this purpose, we post-process both the ground truth and predicted map. Here, we convert the plus annotated ground truth to dot annotated map through a 2D convolution with the kernel, $J = \frac{1}{5} \times K$, i.e., $J = [[0, 0.2, 0], [0.2, 0.2, 0.2], [0, 0.2, 0]]$. After rounding the filtered map, we get the clean dot annotated ground truth. Now, we can

**Table 1**
Summary of the benchmark crowd datasets used for evaluation.

| Dataset | Training images | Test images | Person count | | | Average resolution ($H \times W$) | Crowd density (per megapixel) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Min | Avg | Max | | Minimum | Average | Maximum |
| NWPU-Crowd [8] | 3609 | 1500 | 0 | 412 | 20033 | 2191 × 3209 | 0 | 76 | 2092 |
| UCF-QNRF [4] | 1201 | 334 | 49 | 815 | 12865 | 2011 × 2902 | 3 | 412 | 17292 |
| ShanghaiTech-A [6] | 300 | 182 | 33 | 501 | 3139 | 589 × 868 | 66 | 1111 | 7983 |
| ShanghaiTech-B [6] | 400 | 316 | 9 | 124 | 578 | 768 × 1024 | 11 | 157 | 735 |

calculate the number of persons present in the ground truth map through counting the dots.

To localize persons in the predicted feature map, we perform average pooling and max pooling operations over the same predicted map with 3 × 3 pool size and one stride. After that, we pixel-wise compare both the pooled maps and perform element-wise multiplication as done in [8]. This enables identifying the local peaks efficiently. Additionally, we round the computed feature map such that only strong peaks exist, i.e., the peaks having less than 50% confidence get nullified. Now, we can calculate the number of persons in the predicted map through counting the peaks. If we subtract the predicted count from the true count, we get the localized counting error. Using this custom metric, we save the best model having the lowest validation error during training time. Note that, we use 25% of training images as the validation dataset. Besides, we use Adam algorithm [26] with default parameters as the optimizer for the training purpose.

## 5. Experimental evaluation

In this section, we present experimental evaluation of our LC-Net against state-of-the-art approaches over four benchmark datasets for the crowd localization task. First, we describe our deep learning environment setup used in both training and testing stages.

### 5.1. Environment setup

We use a GPU instance of Amazon Web Service to train and test our models over the datasets. To do so, we launch a Ubuntu 16.04 EC2 P3 instance (p3.2xlarge) hosted in the US East (N. Virginia) [27]. This instance provides up to 10 Gbps of networking throughput, 8 custom Intel Xeon Scalable (Skylake) vCPUs, 1 NVIDIA V100 Tensor Core GPU with 16 GB of memory, and 61 GB main memory. We use the Anaconda environment having TensorFlow 2.3 framework with Python 3.7 and CUDA 10.2 in our experimental evaluation. Using this setup, we evaluate our crowd datasets, which we present next.

### 5.2. Crowd dataset and challenges

We evaluate our LC-Net over four benchmark crowd datasets namely NWPU-Crowd [8], UCF-QNRF [4], ShanghaiTech-A [6], and ShanghaiTech-B [6]. To the best of our knowledge, the NWPU-Crowd and UCF-QNRF are the most recently available comprehensive datasets for crowd localization having high-resolution images and tremendous annotations. The NWPU-Crowd dataset contains 3609 training images and 1500 test images. Besides, the UCF-QNRF contains 1535 images in total, where 132 training images and 29 test images have been collected from Hajj and similar footage. On the other hand, ShanghaiTech-A contains 482 images having congested crowds, which are crawled from the Internet. Besides, ShanghaiTech-B contains 716 low-dense images taken from surveillance cameras in Shanghai. We summarize the characteristics of these datasets in Table 1. Here, it is worth mentioning that the ShanghaiTech datasets contain low-resolution images and the number of training images is much smaller than NWPU-Crowd and UCF-QNRF datasets.

In addition to these datasets, there exists other crowd datasets in the literature such as UCSD [28], Mall [29], UCF-CC-50 [30], Venice [12], etc. However, these datasets have severe limitations to evaluate a crowd localization algorithm. For example, UCSD and Mall datasets contain only a single surveillance scene that lacks data diversity [8]. Besides, UCF-CC-50 and Venice datasets contain 50 and 167 images respectively, which is too small to train a robust deep learning model [8]. On the other hand, our adopted datasets namely NWPU-Crowd, UCF-QNRF, ShanghaiTech-A, and ShanghaiTech-B contain a large number of diversified images. For these reasons, we consider these four datasets leaving the other remaining ones in our performance evaluation.

Although the NWPU-Crowd and UCF-QNRF datasets contain a lot of high-quality images, they create several challenges during the training stage. Next, we present the major two challenges in this regard along with their solutions.

#### 5.2.1. High definition images having sparse crowd

Both NWPU-Crowd and UCF-QNRF datasets contain several high-resolution images sometimes having low crowd density. For example, a particular test image of UCF-QNRF has a resolution of 4912 × 7360, i.e., 36 megapixels, and contains 130 persons resulting in only 4 persons per unit megapixels. If we divide this image into a grid of 512 × 512 pixels, most of the person heads are covered by two or more sub-images. As a result, the contextual information of this image gets lost during patch generation. Besides, we observe that these low-dense but high-resolution images produce low precision and recall resulting in a poor F1 score. Therefore, we need to resize such images according to crowd density during the pre-processing stage. Next, we describe this novel data augmentation technique.

***Density-based image resizing technique***. We pick an image to resize, if the resolution of the image is greater than Full HD (1920 × 1080) and the crowd density is less than 50. Note that, we set this criterion as per our observations. In order to maintain image quality, first, we reduce the resolution of an image to its nearest commercial standard resolution. To be more specific, if the resolution of an image is greater than 4K Ultra HD (3840 × 2160), we reduce the resolution to 4K Ultra HD based on the original aspect ratio. Then, we calculate the crowd density again. If it is still not less than 50, we reduce the resolution to Full HD. We do not further reduce the resolution, since it degrades image quality. Note that, we resize the ground truth feature map of such images accordingly. In this way, we resize the training images of NWPU-Crowd and UCF-QNRF datasets. For testing purpose, we resize all very high definition (greater than 4k Ultra HD) images of UCF-QNRF test set to 4K Ultra HD based on the original aspect ratio, since we do not know the crowd density of test images prior to evaluation.

### 5.2.2. Imbalanced dataset

Most of the sub-images generated from the NWPU-Crowd and UCF-QNRF datasets contain either no people or a few people. These sub-images make the training dataset imbalanced that makes the model biased. Besides, they incur unnecessary computational costs resulting in a much longer training time. In order to reduce the bias, we remove some of the sub-images from the training dataset using a seeded uniform random distribution similar to [4]. To do so, first, we calculate the histogram of people count in the training dataset using Jenks natural breaks [31] as the optimum bin width. After that, we reduce the sub-images of high-frequency bins. Thus, we reduce the bias from the training sub-images of NWPU-Crowd and UCF-QNRF.

## 5.3. Evaluation protocols

We evaluate our trained model using the test datasets and measure its precision, recall, and F1 score. To do so, first, we generate image patches and corresponding ground truth feature maps from the test images similar to training images. Besides, we normalize the test image patches similar to training patches. However, we do not augment the test patches. We feed the image patches to LC-Net and get the predicted feature maps. After that, we post-process the feature maps and count the number of predicted persons. Note that, we accumulate the predicted count for a particular test image from its sub-images. Next, we evaluate the predictions using three state-of-the-art localization protocols namely Euclidean protocol, Gaussian protocol, and adaptive protocol [8]. The basic difference among these protocols is the way to treat a predicted point as a true positive. Next, we describe each of them.

### 5.3.1. Euclidean protocol

Euclidean protocol is proposed by Idrees et al. [4] for UCF-QNRF dataset. It considers the Euclidean distance between predicted and ground truth points. For this purpose, we perform a maximum bipartite matching between the predicted points and ground truth points. Next, we assign each predicted point to a particular true point if they are not previously assigned to others and their distance is less than a particular threshold ($\delta$). In this way, we get a list of true positive points. Dividing the true positives by predicted person count and real person count give precision and recall respectively. The main drawback of this protocol is that it gives the same importance to all predicted points under a particular radius of a ground truth. Hence, a higher confident point may be ignored. It implies that wrong detection can be treated as a true positive due to the radius limit.

### 5.3.2. Gaussian protocol

Liu et al. [7] have proposed the Gaussian protocol to overcome the drawbacks of the Euclidean protocol. In this protocol, first, we sort the predicted points in descending order based on their confidence scores. After that, we classify each predicted point sequentially into a true positive or false positive. A predicted point and its nearest true person will be matched successfully, if they are not matched previously with others and their affinity is greater than a particular threshold. In order to measure the affinity, we impose an un-normalized Gaussian function around each true point using $\sigma$ as the variance. If the Gaussian output of the predicted point, i.e., $f_g(x)$, is greater than a particular threshold, we mark the predicted point as a true positive. Using the true positives, we calculate the precision and recall. The main drawback of this protocol is that it does not consider the scale variation of heads in an image. It imposes the same Gaussian variance to both closer heads and distant heads. However, distant head areas look smaller due to perspective variation in an image.

### 5.3.3. Adaptive protocol

Wang et al. [8] have proposed this scale-aware protocol that considers the scale variation of heads in an image. Similar to the Euclidean protocol, first, we perform a maximum bipartite matching between the predicted points and ground truth points. Next, we assign each predicted point to a particular true point if they are not previously assigned to others and their distance is less than a particular threshold ($\sigma$). For each head with the size of $h \times w$, two thresholds are defined such as $\sigma_s = min(h, w)$ and $\sigma_l = \sqrt{h^2 + w^2}$. In this way, we calculate precision and recall.

## 5.4. Performance evaluation

We evaluate the performance of our proposed LC-Net over four crowd datasets in comparison to existing methods. LC-Net outperforms all the alternative methods over all the datasets based on the F1 score. Note that, we average the results over five iterations for a particular training configuration in the performance evaluation, as the neural network possesses stochastic nature by default. However, the existing methods are yet to consider this stochastic nature. Next, we present the results for each dataset.

### 5.4.1. UCF-QNRF dataset

Table 2 shows the comparative results of LC-Net against three recent methods over UCF-QNRF dataset. From this table, we can see that LC-Net achieves the highest F1 score among all the other methods based on the Euclidean protocol. Note that, the authors of this protocol (Idrees et al. [4]) averaged the localization results over four distance thresholds ($\delta$) ranging from 1 to 100 pixels. However, they did not mention the exact value of these four distance thresholds. Hence, we assume four equal-spaced thresholds within this range, i.e., 20, 40, 60, and 80.

We compare the performance of LC-Net against an existing crowd localization method named as RAZ_Net (Liu et al. [7]) over UCF-QNRF dataset based on Gaussian protocol for a different combination of $\sigma$ and $f_g(x)$. Here, smaller $\sigma$ and higher $f_g(x)$ represents strict threshold condition. The comparison results are shown in Table 3. It confirms that LC-Net achieves the highest performance in terms of F1 score. To be more specific, LC-Net performs better under strict conditions than relaxed ones.

In addition to crowd localization methods, we compare the performance of LC-Net against existing several crowd counting methods on UCF-QNRF dataset. As these methods have no crowd localization ability, we use the pre-processing and post-processing tasks of LC-Net and replace the architectural part only. The comparison results based on Gaussian protocol are shown in Table 4. It confirms that LC-Net achieves the highest localization accuracy. Although MCNN [6] requires a minimum number of computational resources, it exhibits very poor localization accuracy. Among the other methods, LC-Net requires a considerably lower number of computational resources.

### 5.4.2. NWPU-crowd dataset

Table 5 shows the performance comparison of LC-Net against two recent methods over NWPU-Crowd test dataset based on the adaptive protocol. The results have been obtained from the official website of NWPU-Crowd. We can see that LC-Net confirms the best localization performance based on the F1 score. Under a strict threshold, LC-Net achieves better performance than the relaxed threshold.

### 5.4.3. ShanghaiTech Datasets

For ShanghaiTech datasets, we present the performance comparison of LC-Net against RAZ_Net [7] in Table 3. We can see that LC-Net achieves better performance over both ShanghaiTech datasets in most of the cases. As the number of training images in both ShanghaiTech datasets is very low, LC-Net does not attain comprehensive learning to exhibit better test performance [8].

**Table 2**
Performance comparison of LC-Net against existing crowd localization methods on UCF-QNRF dataset based on Euclidean protocol.

| Method | Average Precision (%) | Average Recall (%) | Average F1 Score (%) |
|---|---|---|---|
| Method in [32] | 75.5 | 49.9 | 60.1 |
| MCNN [4,6] | 59.9 | 63.5 | 61.7 |
| LCFCN [33,34] | **77.9** | 52.4 | 62.7 |
| DenseNet63 [4,15] | 70.2 | 58.1 | 63.6 |
| ResNet74 [4,19] | 61.6 | 66.9 | 64.1 |
| CL [4] | 75.8 | 59.8 | 66.8 |
| Encoder–Decoder [4,35] | 71.8 | 63.0 | 67.1 |
| LC-Net | 74.3 (±1.4) | **66.5** (±2.3) | **70.1** (±0.9) |

**Table 3**
Performance comparison of LC-Net against existing crowd localization method over UCF-QNRF, ShanghaiTech-A, and ShanghaiTech-B datasets based on Gaussian protocol.

| Dataset | $\sigma$ | Method | $f_g(x) >= 0.5$ | | | $f_g(x) >= 0.75$ | | |
|---|---|---|---|---|---|---|---|---|
| | | | Average Precision (%) | Average Recall (%) | Average F1 Score (%) | Average Precision (%) | Average Recall (%) | Average F1 Score (%) |
| UCF-QNRF | 5 | RAZ_Net [7] | 7.9 | 24.2 | 11.9 | 3.1 | 14.3 | 5.1 |
| | | LC-Net | 42.2 (±1.2) | 41.2 (±0.5) | 41.7 (±0.4) | 27.0 (±0.8) | 26.6 (±0.2) | 26.8 (±0.3) |
| | 20 | RAZ_Net [7] | 41.4 | 60.2 | 49.1 | 28.7 | 49.7 | 36.4 |
| | | LC-Net | 70.5 (±1.5) | 65.1 (±2.0) | 67.6 (±0.6) | 63.4 (±1.5) | 59.4 (±1.6) | 61.3 (±0.4) |
| | 40 | RAZ_Net [7] | 57.3 | 71.9 | 63.8 | 48.1 | 65.2 | 55.4 |
| | | LC-Net | 71.3 (±1.5) | 65.6 (±2.1) | 68.3 (±0.7) | 66.7 (±1.5) | 61.6 (±1.9) | 64.0 (±0.6) |
| Shanghai-Tech-A | 5 | RAZ_Net [7] | 36.0 | 40.9 | 38.3 | 20.5 | 57.9 | 30.3 |
| | | LC-Net | 62.1 (±0.8) | 65.5 (±1.0) | 63.7 (±0.4) | 46.7 (±0.6) | 49.2 (±1.1) | 47.9 (±0.6) |
| | 20 | RAZ_Net [7] | 66.7 | 79.9 | 72.7 | 60.1 | 75.3 | 66.9 |
| | | LC-Net | 77.9 (±1.3) | 81.3 (±0.9) | 79.5 (±0.4) | 73.0 (±1.2) | 76.3 (±0.9) | 74.6 (±0.2) |
| | 40 | RAZ_Net [7] | 74.5 | 84.7 | 79.3 | 69.9 | 82.0 | 75.5 |
| | | LC-Net | 78.4 (±1.4) | 81.8 (±0.9) | 80.0 (±0.4) | 74.1 (±1.3) | 77.2 (±0.8) | 75.6 (±0.3) |
| Shanghai-Tech-B | 5 | RAZ_Net [7] | 45.6 | 66.1 | 53.4 | 28.1 | 51.6 | 36.4 |
| | | LC-Net | 64.2 (±0.6) | 66.6 (±1.1) | 65.4 (±0.5) | 47.5 (±0.5) | 49.1 (±0.9) | 48.3 (±0.6) |
| | 20 | RAZ_Net [7] | 68.7 | 82.0 | 74.8 | 64.9 | 79.4 | 71.4 |
| | | LC-Net | 80.7 (±0.9) | 83.8 (±1.3) | 82.2 (±0.5) | 74.6 (±0.8) | 77.4 (±1.2) | 76.0 (±0.4) |
| | 40 | RAZ_Net [7] | 75.3 | 85.7 | 80.2 | 71.6 | 83.4 | 77.1 |
| | | LC-Net | 81.0 (±0.9) | 84.0 (±1.3) | 82.5 (±0.4) | 73.4 (±0.8) | 76.0 (±1.1) | 74.7 (±0.3) |

**Table 4**
Performance comparison of LC-Net against existing crowd counting methods on UCF-QNRF dataset based on Gaussian protocol.

| Method | $\sigma = 20, f_g(x) >= 0.5$ | | | $\sigma = 40, f_g(x) >= 0.5$ | | | GFLOPS | FPS (K80) | Size (MB) |
|---|---|---|---|---|---|---|---|---|---|
| | Average Pre (%) | Average Rec (%) | Average F1 (%) | Average Pre (%) | Average Rec (%) | Average F1 (%) | | | |
| MCNN [6] | 52.8 (±2.0) | 57.2 (±3.9) | 54.8 (±1.3) | 54.1 (±2.0) | 58.6 (±4.1) | 56.1 (±1.4) | 17.7 | 1.18 | 0.7 |
| VGG-19 [18] | 68.1 (±2.4) | 61.0 (±1.5) | 64.3 (±1.7) | 68.8 (±2.3) | 61.6 (±1.5) | 65.0 (±1.6) | 378.2 | 0.16 | 86.6 |
| CSRNet [5] | 68.1 (±2.0) | 61.6 (±3.4) | 64.7 (±2.3) | 69.0 (±2.0) | 62.2 (±3.4) | 65.4 (±2.4) | 320.2 | 0.25 | 65.3 |
| LC-Net | **70.5** (±1.5) | **65.1** (±2.0) | **67.6** (±0.6) | **71.3** (±1.5) | **65.6** (±2.1) | **68.3** (±0.7) | 118.9 | 0.30 | 19.3 |

**Table 5**
Performance comparison of LC-Net against existing crowd localization methods on NWPU-Crowd dataset based on adaptive protocol.

| Method | $\sigma = s$ | | | $\sigma = l$ | | |
|---|---|---|---|---|---|---|
| | Precision (%) | Recall (%) | F1 Score (%) | Precision (%) | Recall (%) | F1 Score (%) |
| Faster RCNN [8,36] | **89.4** | 3.3 | 6.3 | **95.8** | 3.5 | 6.7 |
| VGG+GPR [8,37] | 45.3 | 40.2 | 42.6 | 55.8 | 49.6 | 52.5 |
| FPN [33] | 57.1 | 46.1 | 51.0 | 65.9 | 53.3 | 58.9 |
| RAZ_Loc [7,8] | 57.6 | 47.0 | 51.7 | 66.6 | 54.3 | 59.8 |
| TinyFaces [8,38] | 49.1 | **56.6** | 52.6 | 52.9 | **61.1** | 56.7 |
| AutoScale [33] | 59.1 | 50.4 | 54.4 | 67.3 | 57.4 | 62.0 |
| LC-Net | 61.3 | 56.4 | **58.7** | 65.6 | 60.3 | **62.8** |

#### 5.4.4. Ablation studies

We present the ablation study of our proposed LC-Net over UCF-QNRF dataset in Table 6. It confirms the effectiveness of both residual layers and dilated convolution in our proposed network. Besides, the impact of density-based image resizing technique on the performance of LC-Net over NWPU-Crowd validation and UCF-QNRF test datasets is shown in Table 7. It also confirms the effectiveness of our proposed data augmentation technique. Here, the evaluation is based on the adaptive protocol for NWPU-Crowd and Euclidean protocol for UCF-QNRF respectively.

### 5.5. Qualitative analysis

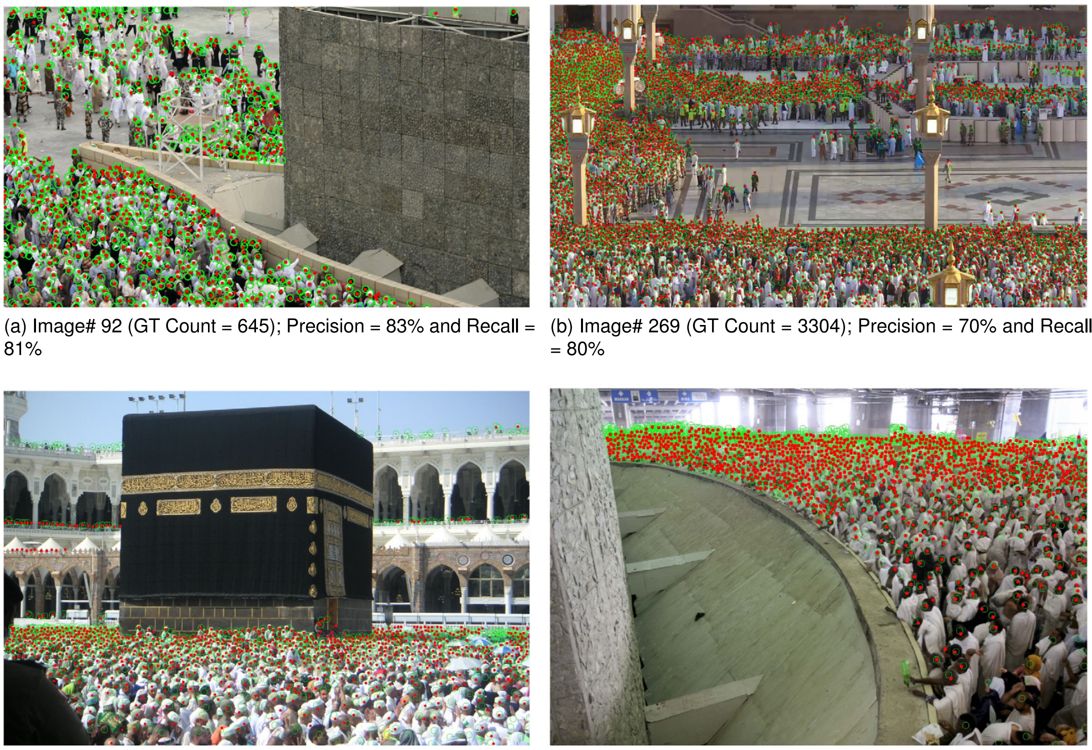
Fig. 3 presents the predictions of LC-Net over four test images of UCF-QNRF dataset. Here, we denote a ground truth using a green circle and a predicted location using a red point. The center of a green circle is the actual ground truth location and the radius of that circle is $\delta$. If a predicted point falls within the circular area of a ground truth location, we treat the predicted point as a true positive. Thus, the figures demonstrate that our proposed LC-Net localizes the persons with high precision under a strict threshold.

**Table 6**
Ablation studies of LC-Net over UCF-QNRF dataset based on Gaussian protocol.

| Residual layers | Dilated convolution | $\sigma = 20, f_g(x) >= 0.5$ | | | $\sigma = 40, f_g(x) >= 0.5$ | | |
|---|---|---|---|---|---|---|---|
| | | Average Precision (%) | Average Recall (%) | Average F1 Score (%) | Average Precision (%) | Average Recall (%) | Average F1 Score (%) |
| No | No | 68.1 ($\pm$3.6) | 61.9 ($\pm$2.3) | 64.8 ($\pm$1.4) | 69.0 ($\pm$3.6) | 62.5 ($\pm$2.4) | 65.5 ($\pm$1.4) |
| No | Yes | 69.3 ($\pm$2.8) | 61.3 ($\pm$1.9) | 65.0 ($\pm$0.6) | 70.2 ($\pm$2.7) | 62.1 ($\pm$2.0) | 65.8 ($\pm$0.6) |
| Yes | No | 70.4 ($\pm$3.3) | 64.2 ($\pm$1.3) | 67.1 ($\pm$1.9) | 71.2 ($\pm$3.4) | 64.8 ($\pm$1.4) | 67.8 ($\pm$1.9) |
| Yes | Yes | **70.5** ($\pm$1.5) | **65.1** ($\pm$2.0) | **67.6** ($\pm$0.6) | **71.3** ($\pm$1.5) | **65.6** ($\pm$2.1) | **68.3** ($\pm$0.7) |

**Table 7**
Impact of density-based image resizing technique on the performance of LC-Net on NWPU-Crowd validation and UCF-QNRF test datasets based on adaptive and Euclidean protocols respectively.

| Dataset | Density-based image resizing | $\sigma$ | Precision (%) | Recall (%) | F1 Score (%) | $\sigma$ | Precision (%) | Recall (%) | F1 Score (%) |
|---|---|---|---|---|---|---|---|---|---|
| NWPU-Crowd | No | $s$ | 54.4 | 57.7 | 56.0 | $l$ | 58.3 | 61.8 | 60.0 |
| | Yes | | **63.3** | **60.5** | **61.9** | | **67.4** | **64.5** | **65.9** |
| UCF-QNRF | No | 20 | 66.0 | 56.9 | 61.1 | 40 | 70.4 | 59.7 | 64.6 |
| | Yes | | **70.0** | **63.6** | **66.6** | | **74.3** | **66.5** | **70.2** |


(a) Image# 92 (GT Count = 645); Precision = 83% and Recall = 81%

(b) Image# 269 (GT Count = 3304); Precision = 70% and Recall = 80%

(c) Image# 215 (GT Count = 1048); Precision = 74% and Recall = 69%

(d) Image# 93 (GT Count = 1929); Precision = 87% and Recall = 56%

**Fig. 3.** Qualitative analysis over four test images in UCF-QNRF dataset, where a green circle represents the circular area around a ground truth location (radius, $\delta = 10$) as per Euclidean protocol and a red point represents a predicted location.

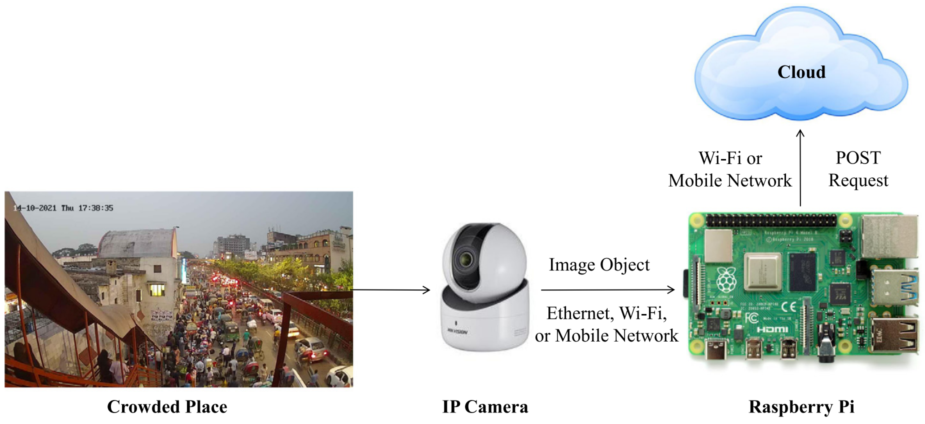
## 6. Real implementation

In this section, we discuss a real implementation of our proposed approach using a client–server application. For the client end, we develop a crowd image capturing module that uploads the captured images to a web server. In the server end, we develop a website to execute our crowd localization algorithm and show the results in real-time. Next, we describe each of the development phases in detail.

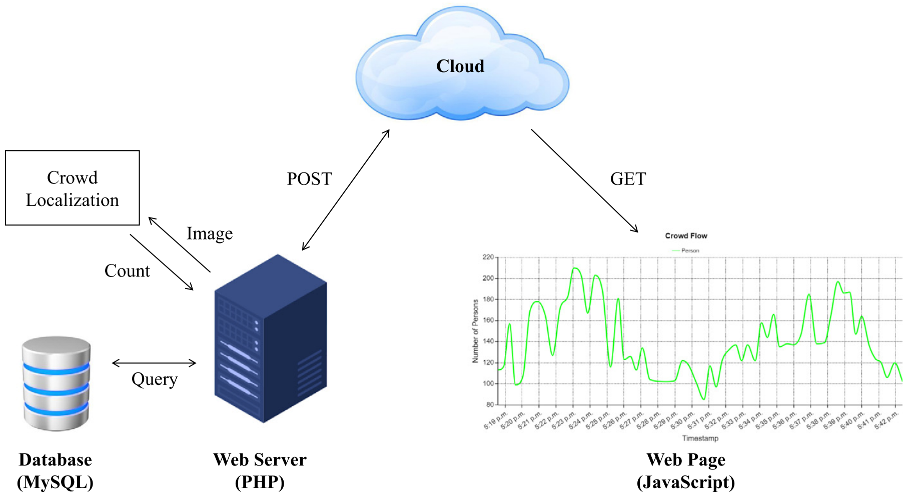
### 6.1. Development of image capturing module

In our image capturing module, we use a Hikvision 1.0 MP (1280 $\times$ 720) IP camera that has Pan–Tilt–Zoom features, night vision capability, and Internet connectivity [39]. It sends the captured images to a Raspberry Pi using a wireless network. Here, we use a Raspberry Pi 4 Model B that has 4 Cortex-A72 (ARMv7) CPUs (1.5 GHz) and 4 GB main memory [40]. The Raspberry Pi uploads the captured image to the cloud (web server) using a wireless network. We show the architecture of this image capturing module in Fig. 4(a).

We use Python programming language to implement the functionalities of the capturing module. Here, we send the captured image along with its time-stamp through a POST request to the server. To ensure secure data transmission, we send a personal access token through the header of the POST request that protects the server from intruders. Note that, the access token is provided by our web application for a particular capturing module.



(a) Architecture of our crowd image capturing module



(b) Architecture of the web hosting system running our web application

**Fig. 4.** Real implementation of our proposed approach.

### 6.2. Development of web application

To perform the crowd monitoring tasks, we develop a web application that receives the crowd images from the capturing module. After that, it runs the LC-Net model over the captured crowd images and generates the localized crowd counting values. Finally, it stores the generated values and visualizes them in a graphical manner. We show the architecture of this web application in Fig. 4(b).

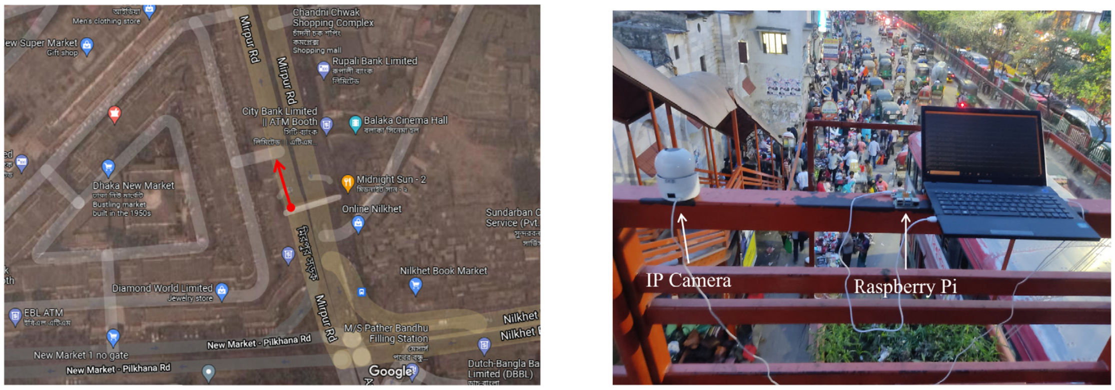
To implement these functionalities, we use PHP-based Laravel 8 framework [41] that implements conventional model–view–controller architecture for web services. We build a Laravel API to receive the data sent by Raspberry Pi, run the LC-Net model, and store the information in a MySQL database.

For end-users, we design a web interface through using CSS-based Tailwind framework [42] and JavaScript-based Chart.js library [43]. In this interface, we present two charts on crowd flow and estimation time. The chart on crowd flow shows the localized crowd counting values as per the time-stamps of image capturing. Besides, the chart on estimation time shows the inference time and total time required for performing the crowd localization tasks. Here, inference time refers to the time required to get the localized counting values from LC-Net. On the other hand, the total time refers to the time duration from the image capturing to database insertion, which involves all the processing on the image. In short, the total time is the sum of capturing time, inference time, and network latency.

We host our developed web application on a web server. Our web server has two Intel Xeon E5-2680 CPUs (2.40 GHz) and 8 GB main memory. Note that, our web server has no GPU. Our web application can be accessed through this link: http://nec.cse.buet.ac.bd/people.

**Table 8**
Data collection scenario for real-world evaluation.

| Attribute | Online evaluation |
|---|---|
| Location | New Market Area, Dhaka, Bangladesh |
| Date | October 14, 2021 |
| Time | 5:18 PM - 5:41 PM |
| Duration | 23 min |
| # of images | 57 |



(a) Location of data collection    (b) Snapshot of real deployment

**Fig. 5.** Crowd data collection at the entrance of New Market Area, Dhaka, Bangladesh.

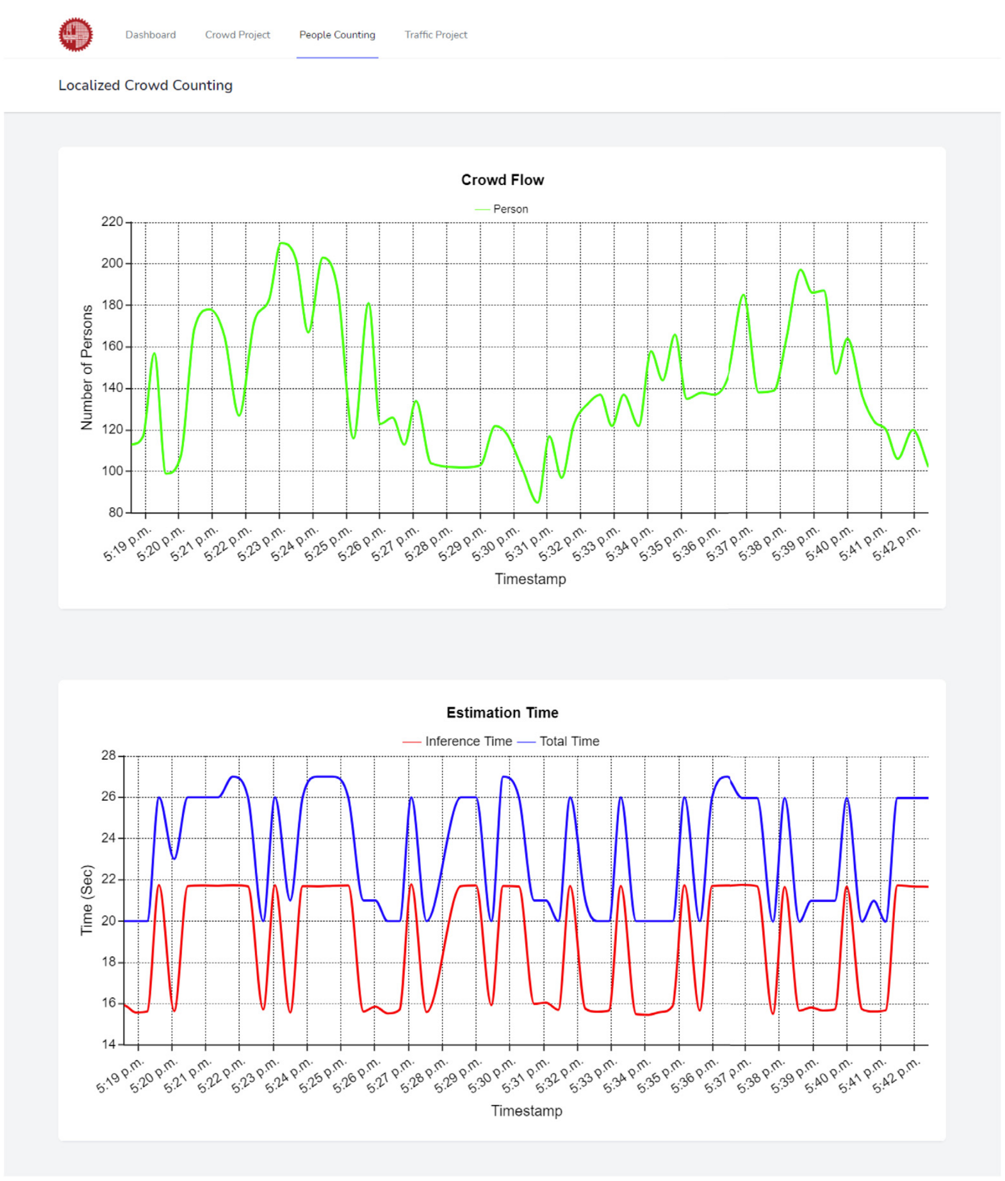
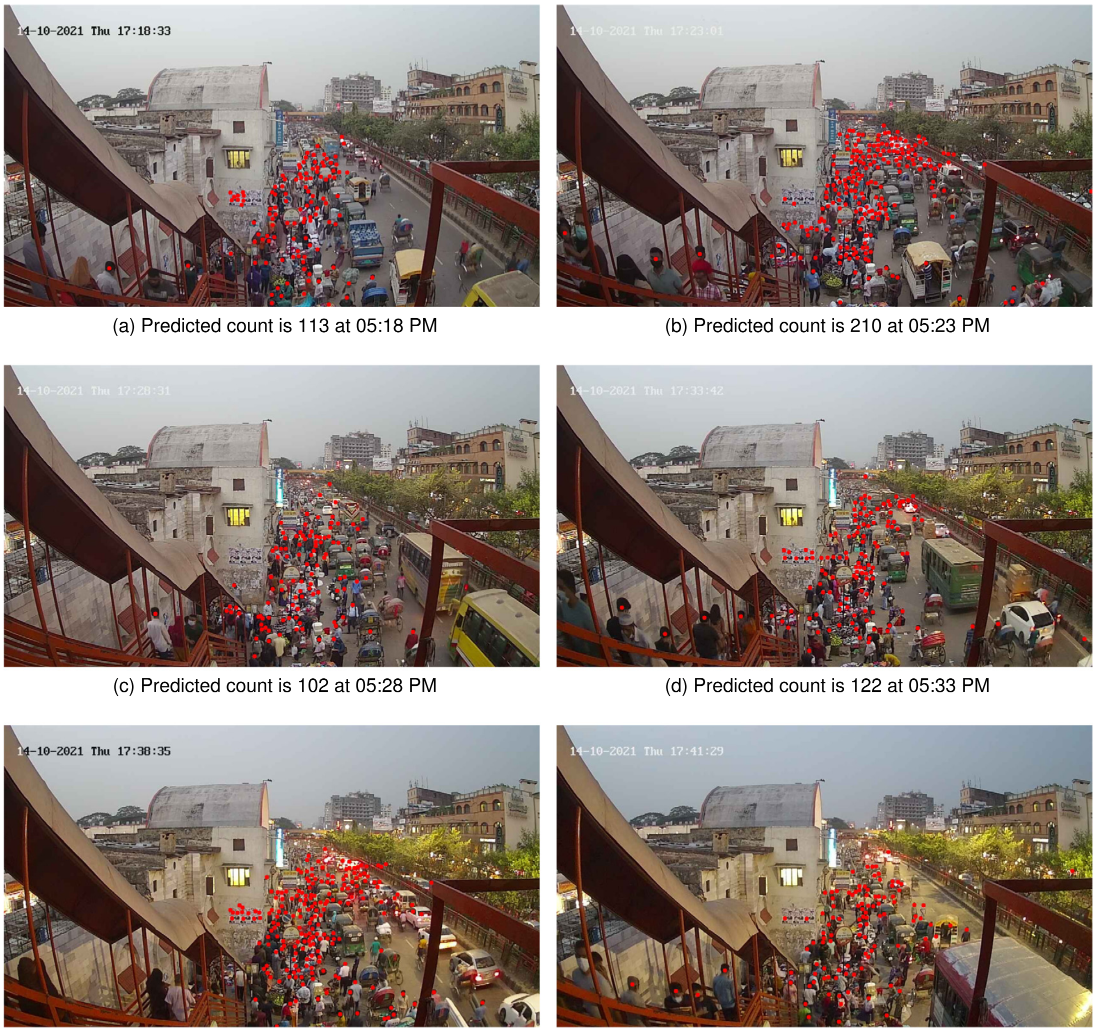


**Fig. 6.** Crowd counting results at the entrance of New Market Area, Dhaka, Bangladesh.



(a) Predicted count is 113 at 05:18 PM

(b) Predicted count is 210 at 05:23 PM

(c) Predicted count is 102 at 05:28 PM

(d) Predicted count is 122 at 05:33 PM

(e) Predicted count is 197 at 05:38 PM

(f) Predicted count is 106 at 05:41 PM

**Fig. 7.** Outputs of real-time crowd localization using LC-Net at New Market Area, Dhaka, Bangladesh.

## 6.3. *Real-time evaluation*

To evaluate the complete system in real-time, we setup our image capturing module on a foot-over bridge at the New Market Area in Dhaka, capital of Bangladesh (a developing country of southern Asia). We collect crowd images from the entrance of this market on October 14, 2021, from 5:18 PM to 5:41 PM. We present the details of our data collection in Table 8. Fig. 5 shows the location of data collection and a snapshot of real deployment. Here, our client module captures a crowd image and uploads it to the server over an available mobile network (3G/4G). Here, we use a laptop as the power source for both Raspberry Pi and IP Camera.

In the server, we run the LC-Net model (trained over UCF-QNRF [4] dataset) on the captured images. From the server, we get the charts on crowd flow and estimation time as shown in Fig. 6. Here, we can see that the number of persons is varying over time. From the time chart, we can see that most of the images require 16–21 s for performing inference tasks. The reason behind such high inference time is that there is no GPU available on our web server. Our system requires 20–26 s for performing full operation covering both inference and communication to the server. Here, the time difference between the total time and inference time represents model initialization time and network latency. Note that, the network latency is introduced by the image transmission between IP Camera to Raspberry Pi and Raspberry Pi to server.

We show six example output images of our real-time crowd localization in Fig. 7. Here, we can see that the number of predicted persons is varying according to the density of the crowd as shown in Fig. 6. Besides, these output images show that LC-Net can pinpoint the persons in most of the cases.

From our experimental evaluation, we can see that LC-Net localizes the people more accurately and quickly than state-of-the-art approaches. It implies that LC-Net can provide a more accurate initial state for a crowd tracking algorithm in an efficient manner, which is required to estimate the crowd flow in a particular event. Note that, the crowd flow estimation tasks are required for allocating appropriate manpower and other resources to ensure public safety and security [9]. Therefore, our proposed LC-Net and its deployment similar to what we show in this section can facilitate the tasks of monitoring authority in making decisions for crowd management.

## 7. Conclusion and future work

Crowd management in case of large mass gatherings requires careful monitoring to ensure public safety and security that can be facilitated by visual crowd analysis tasks such as crowd counting, localization, and tracking. Existing crowd counting work adopt a density map-based approach to count the persons with minimal error, however, these approaches lack spatial information of the people that often makes them unusable in advanced crowd analysis tasks such as crowd tracking, crowd flow estimation, etc. Hence, we propose a novel deep learning architecture named LC-Net through exploiting residual layers and dilated convolution for precise crowd localization in an efficient manner. Experimental evaluation of LC-Net shows a substantial performance improvement over four different crowd benchmark datasets compared to state-of-the-art alternatives.

For implementing our proposed deep learning architectures in the real-world, we propose a client–server application using a Raspberry Pi (as a client) and a web server. The Raspberry Pi captures the crowd images and uploads them to the web server. In the server, we perform crowd localization tasks over captured images and visualize the counting values in a website. This client–server demonstrates the applicability of our proposed approach.

The strength of our approach is that it can pinpoint medium-scale heads more accurately from the highly congested scenarios. However, it cannot perform well in the case of too small-scale or too large-scale heads. To handle this scale variation, we plan to develop a multi-scale localization network in the future. Besides, we will devise and integrate a crowd tracking module to predict the flow of the crowd.

## CRediT authorship contribution statement

**Tarik Reza Toha:** Conceptualization, Methodology, Investigation, Software, Writing – original draft. **Najla Abdulrahman Al-Nabhan:** Project administration, Funding acquisition. **Saiful Islam Salim:** Data curation. **Masfiqur Rahaman:** Data curation, Resources. **Uday Kamal:** Validation, Conceptualization. **A.B.M. Alim Al Islam:** Supervision, Conceptualization, Writing – review and editing, Project administration.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (DRI-KSU-762).

## References

[1] A.S. Alqahtani, K.E. Wiley, M. Tashani, H.W. Willaby, A.E. Heywood, N.F. BinDhim, R. Booy, H. Rashid, Exploring barriers to and facilitators of preventive measures against infectious diseases among Australian hajj pilgrims: Cross-sectional studies before and after hajj, Int. J. Infect. Dis. 47 (2016) 53–59.
[2] GASTAT, Hajj Statistics 2019 - 1440, Tech. rep., General Authority for Statistics, Kingdom of Saudi Arabia, 2019, URL https://www.stats.gov.sa/en/28. (Accessed 04 May 2021).
[3] BBC, Hajj stampede: At least 717 killed in Saudi Arabia, 2015, https://www.bbc.com/news/world-middle-east-34346449. (Accessed 04 May 2021).
[4] H. Idrees, M. Tayyab, K. Athrey, D. Zhang, S. Al-Maadeed, N. Rajpoot, M. Shah, Composition loss for counting, density map estimation and localization in dense crowds, in: Proceedings of the 15th European Conference on Computer Vision, ECCV, Springer, Munich, Germany, 2018, pp. 544–559.
[5] Y. Li, X. Zhang, D. Chen, CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes, in: Proceedings of the 31st Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Salt Lake City, UT, USA, 2018, pp. 1091–1100.
[6] Y. Zhang, D. Zhou, S. Chen, S. Gao, Y. Ma, Single-image crowd counting via multi-column convolutional neural network, in: Proceedings of the 29th Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Las Vegas, NV, USA, 2016, pp. 589–597.
[7] C. Liu, X. Weng, Y. Mu, Recurrent attentive zooming for joint crowd counting and precise localization, in: Proceedings of the 32nd Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Long Beach, CA, USA, 2019, pp. 1217–1226.
[8] Q. Wang, J. Gao, W. Lin, X. Li, NWPU-crowd: A large-scale benchmark for crowd counting and localization, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 43 (6) (2021) 2141–2149.
[9] S.D. Khan, M. Tayyab, M.K. Amin, A. Nour, A. Basalamah, S. Basalamah, S.A. Khan, Towards a Crowd Analytic Framework For Crowd Management in Majid-al-Haram, Tech. Rep. 1709.05952, arXiv, 2017, URL https://arxiv.org/abs/1709.05952.
[10] M. Tan, R. Pang, Q.V. Le, EfficientDet: Scalable and efficient object detection, in: Proceedings of the 33th International Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Seattle, WA, USA, 2020, pp. 10778–10787.
[11] V.A. Sindagi, 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.
[12] W. Liu, M. Salzmann, P. Fua, Context-aware crowd counting, in: Proceedings of the 32nd Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Long Beach, CA, USA, 2019, pp. 5094–5103.
[13] H. Xiong, H. Lu, C. Liu, L. Liu, Z. Cao, C. Shen, From open set to closed set: Counting objects by spatial divide-and-conquer, in: Proceedings of the 21st International Conference on Computer Vision, ICCV, IEEE, Seoul, South Korea, 2019, pp. 8361–8370.
[14] Z. Ma, X. Wei, X. Hong, Y. Gong, BayesIan loss for crowd count estimation with point supervision, in: Proceedings of the 21st International Conference on Computer Vision, ICCV, IEEE, Seoul, South Korea, 2019, pp. 6141–6150.
[15] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the 30th International Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Honolulu, HI, USA, 2017, pp. 2261–2269.
[16] L. Zhu, R. Deng, M. Maire, Z. Deng, G. Mori, P. Tan, Sparsely aggregated convolutional networks, in: Proceedings of the 15th European Conference on Computer Vision, ECCV, Springer, Munich, Germany, 2018, pp. 192–208.
[17] J. Redmon, A. Farhadi, YOLOv3: An Incremental Improvement, 2018, arXiv URL https://arxiv.org/abs/1804.02767.
[18] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, 2015, arXiv URL https://arxiv.org/abs/1409.1556.
[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the 29th Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Las Vegas, NV, USA, 2016, pp. 770–778.
[20] J. Redmon, A. Farhadi, YOLO9000: Better, faster, stronger, in: Proceedings of the 30th Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Honolulu, HI, USA, 2017, pp. 6517–6525.
[21] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning, vol. 37, ICML, PMLR, Lille, France, 2015, pp. 448–456, URL http://proceedings.mlr.press/v37/ioffe15.html. (Accessed 04 May 2021).
[22] A.L. Maas, A.L. Maas, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proceedings of the 30th International Conference on Machine Learning, vol. 28, ICML, JMLR, Atlanta, Georgia, USA, 2013, pp. 448–456, URL http://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.
[23] G. Bradski, The opencv library, Dr Dobb's J. Softw. Tools 25 (2000) 120–125.
[24] F. Chollet, et al., Keras, 2020, https://keras.io. (Accessed 04 May 2021).
[25] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, in: Proceedings of the 17th International Conference on Computer Vision, ICCV, IEEE, Santiago, Chile, 2015, pp. 1026–1034.
[26] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015, URL http://arxiv.org/abs/1412.6980.
[27] AWS, Amazon EC2 P3 instances, 2015, https://aws.amazon.com/ec2/instance-types/p3/. (Accessed 04 May 2021).
[28] A.B. Chan, Z.-S.J. Liang, N. Vasconcelos, Privacy preserving crowd monitoring: Counting people without people models or tracking, in: Proceedings of the 21st Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Anchorage, AK, USA, 2008, pp. 1–7.

[29] K. Chen, C.C. Loy, S. Gong, T. Xiang, Feature mining for localised crowd counting, in: Proceedings of the British Machine Vision Conference, BMVA Press, Surrey, England, UK, 2012, pp. 21.1–21.11, URL http://dx.doi.org/10.5244/C.26.21.

[30] H. Idrees, I. Saleemi, C. Seibert, M. Shah, Multi-source multi-scale counting in extremely dense crowd images, in: Proceedings of the 26th Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Portland, OR, USA, 2013, pp. 2547–2554, http://dx.doi.org/10.1109/CVPR.2013.329.

[31] G.F. Jenks, The data model concept in statistical mapping, in: International Yearbook of Cartography, vol. 7, 1967, pp. 186–190.

[32] J. Ribera, D. Güera, Y. Chen, E.J. Delp, Locating objects without bounding boxes, in: Proceedings of the 32nd Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Long Beach, CA, USA, 2019, pp. 6472–6482, http://dx.doi.org/10.1109/CVPR.2019.00664.

[33] C. Xu, D. Liang, Y. Xu, S. Bai, W. Zhan, X. Bai, M. Tomizuka, AutoScale: Learning to Scale for Crowd Counting and Localization, Tech. Rep. 1912.09632, arXiv, 2021, URL https://arxiv.org/abs/1912.09632.

[34] I.H. Laradji, N. Rostamzadeh, P.O. Pinheiro, D. Vazquez, M. Schmidt, Where are the blobs: Counting by localization with point supervision, in: Proceedings of the 15th European Conference on Computer Vision, ECCV, Springer, Munich, Germany, 2018, pp. 560–576, http://dx.doi.org/10.1007/978-3-030-01216-8_34.

[35] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 39 (12) (2017) 2481–2495, http://dx.doi.org/10.1109/TPAMI.2016.2644615.

[36] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 39 (6) (2017) 1137–1149, http://dx.doi.org/10.1109/TPAMI.2016.2577031.

[37] J. Gao, T. Han, Q. Wang, Y. Yuan, Domain-Adaptive Crowd Counting Via Inter-Domain Features Segregation and Gaussian-Prior Reconstruction, Tech. Rep. 1912.03677, 2019, arXiv URL https://arxiv.org/abs/1912.03677.

[38] P. Hu, D. Ramanan, Finding tiny faces, in: Proceedings of the 30th Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Honolulu, HI, USA, 2017, pp. 1522–1530, http://dx.doi.org/10.1109/CVPR.2017.166.

[39] Star Tech Engineering Ltd., Hikvision DS-2CV2Q01EFD-IW (1.0mp) mini PT dome IP camera, 2022, https://www.startech.com.bd/hikvision-ds-2cv2q01efd-iw-ip-camera. (Accessed 23 March 2022).

[40] Raspberry Pi, Raspberry pi 4 tech specs, 2021, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. (Accessed 17 October 2021).

[41] T. Otwell, et al., Laravel: A PHP framework for web artisans, 2022, https://github.com/laravel/laravel. (Accessed 23 March 2022).

[42] A. Wathan, et al., Tailwind CSS: A utility-first CSS framework for rapid UI development, 2022, https://github.com/tailwindlabs/tailwindcss. (Accessed 23 March 2022).

[43] E. Timberg, et al., Chart.js: Simple HTML5 charts using the canvas tag, 2022, https://github.com/chartjs/Chart.js. (Accessed 23 March 2022).