EARLY DETECTION AND PREVENTION OF ANOMALIES USING MCNN

T. Suvarna Kumari
Assistant Professor, Department of CSE,CBIT, Hyderabad.
EMAILID: suvarnakumarit@gmail.com

ABSTRACT: Crowd detection and density estimation from crowded images have a wide range of applications such as crime detection, congestion, public safety, crowd abnormalities, visual surveillance and urban planning. CNN based techniques helps to estimate the crowd detection count and density. The job of detecting a face in the crowd is complicated due to its variability present in human faces including color, pose, expression, position, orientation, and illumination. The proposed approach is simple but effective Multi-column Convolutional Neural Network (MCNN) architecture to map the image to its crowd density map. The proposed MCNN allows the input image to be of arbitrary size or resolution. By utilizing filters with receptive fields of different sizes, the features learned by each column CNN are adaptive to variations in people/head size due to perspective effect or image resolution.

Keywords: Convolutional Neural Network (CNN), Multi-column Convolutional Neural Network (MCNN), Crowd Detection

1. INTRODUCTION

In recent years, the human population is growing in extreme rate hence the growth has indirectly increased the incidence of the crowd. There is a lot of interest in much scientific research in public service, security, safety and computer vision for the analysis of mobility and behavior of the crowd. The job of detecting a face in the crowd is complicated because of showing variance human faces including color, pose, expression, position, orientation, and illumination. Crowd counting plays an important role in urban management and public security. Recently, deep learning has shown a great advantage in making the quality of crowd counting more accurate. However, how to apply deep learning models to embedded terminals is still a challenging issue. In order to achieve the crowd counting by edge computing, we propose a tiny model based on convolutional neural networks to reduce above mentioned situations.

Due to a crowded crisis, there are large crowds of confusion, consequence in pushing, mass-panic, stampede or crowd crushes and causing control loss. Some examples of crowd tragedies, crushes, stampede, like Heavy rains killed 22 people and injured hundreds of others in the afternoon between Mumbai, Parel and Elphinstone Road 2017, 27 pedestrians died due to a stampede on the banks of Godavari river 2015 in southern Indian state of Andhra Pradesh a people died and 26 others injured after the Stampede held on the occasion of Diwali at Gandhi Maidan 2014.

To prevent these fatalities, automatically detection of critical and unusual situations in the dense crowd is necessary. Calculating the number and density of a crowd of people is one of the fundamental requirements in urban management, public security, and business activities. Crowd counting based on computer vision can use image or video data to estimate the crowd counts and density distribution,

which works more conveniently compared with traditional gate counting manners. Focusing on this application, many attempts have been made, such as using histograms of oriented gradient (HOG) descriptors and linear SVM (Support Vector Machine) to detect human bodies [, or using Markov Random Field (MRF) based change detection map and linear regression to obtain estimated count of the crowd . With the development of deep learning technology, many models based on convolutional neural networks (CNNs) have been designed for crowd counting and got remarkable estimation results. However, existing models generally requires too much memory and power consumption in the computation, This mode needs to transmit the image/video data from terminal cameras into the computing server, which requires a building of computing server and large amounts of communication bandwidth. It usually confronts difficulties in practical applications.

The proposed system is simple but effective Multi-column Convolutional Neural Network (MCNN) architecture to map the image to its crowd density map. The proposed MCNN allows the input image to be of arbitrary size or resolution. By utilizing filters with receptive fields of different sizes, the features learned by each column CNN are adaptive to variations in people/head size due to perspective effect or image resolution. Furthermore, the true density map is computed accurately based on geometry-adaptive kernels which do not need knowing the perspective map of the input image. Since exiting crowd counting datasets do not adequately cover all the challenging situations considered in our work, we have collected and labelled a large new dataset that includes 1198 images with about 330,000 heads annotated. On this challenging new dataset, as well as all existing datasets, we conduct extensive experiments to verify the effectiveness of the proposed model and method. In particular, with the proposed simple MCNN model, our method outperforms all existing methods. To estimate the number of people in a given image via the Convolutional Neural Networks (CNNs), there are two natural configurations. One is a network whose input is the image and the output is the estimated head count. The other one is to output a density map of the crowd (say how many people per square meter), and then obtain the head count by integration. Due to perspective distortion, the images usually contain heads of very different sizes; hence filters with receptive fields of the same size are unlikely to capture characteristics of crowd density at different scales.

2. LITERATURE REVIEW

Different approaches are there in Crowd Detection System consists of Input Data, Approaches, Features and Conclusion. There are various approaches have been taken to handle the problem which can be broadly divided into detection based approaches, regression based approaches, density based approaches.

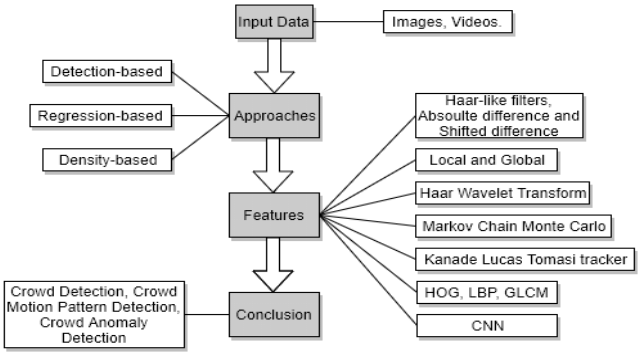

Figure 1. General Structure of Crowd Detection System

2.1 Detection-based Approaches:

Detection model tries to determine the number of people by identifying a single person and their places at the same time. Here we use a moving window like detector to identify people in an image

and count how any there are using HOG descriptors and linear SVM to detect human. By reason of not detecting all the low level features of serious impact from occlusion, some researches were did on the detection of partial bodies. In [4] used AdaBoost to select features and detect face targets by extracting Haar-like features. But this method failed in case of when the features covered by other objects. Other Implementations, a Kanade Lucas Tomasi (KLT) tracker which is highly parallelized which used to extract an expensive arrangement of low-level features to identify the object which is in moving the state from the scene.

2.2 Regression based Approaches:

It is unable to extract low level features using the above. Regression based approach works on local image patches which extract mapping between features for counting purpose. There are various features to encode low-level information such as foreground features, edge features texture and gradient features.

These methods capture local and global properties of the scene such as Local Binary Pattern (LBP), Histogram of Oriented Gradients (HOG), Gray Level Cooccurrence Matrices (GLCM) to improve results. After extracting local and global features, different regression techniques are applied such as linear regression, ridge regression, the neural network to learn to map for crowd counting purpose [2]. In paper [2] extracted MRF based change map as features and then used linear regression to obtain the estimated amount of people. In [8] used textures extracted from gray-level dependence matrices, straight line segments, Fourier analysis, and fractal dimensions in their method, where three types of classifiers, neural, Bayesian and fitting-function, were used to form an effective model for crowd counting.

2.3 CNN Based Approaches

Convolutional Neural Network (CNN) is now widely used in computer vision. Compared with traditional methods, CNN is more capable of dealing with complex scenes, such as severe occlusions and dramatic changes in perspectives. In recent years, there have been many attempts on CNN-based crowd counting algorithms. For example, In [4] used AlexNet [5] and changed the last 4,096 neurons into one neuron to represent the number of people. In [18] focused on designing a network that is convenient for fine-tuning the trained model to adapt to the new monitoring scenario. The authors also used a new density map generation method with perspective information, which was proved better for the training network with alternative regressions of count numbers and density map

In all, these above-mentioned models are designed for server-side computing. The high cost of computation makes most of them difficult to be deployed in embedded devices. Furthermore, fixed input image or model size also makes these models unable to meet variable deployments and speed requirements. In this paper, an innovative model is designed to solve the existing problems for embedded terminals.

3. METHODOLOGY

There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering. Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then

overlaps with systems analysis, systems engineering and systems architecture

.

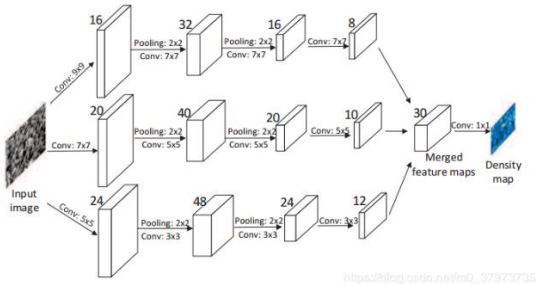

Figure 2. Architectural Overview of Proposed System

Proposed Algorithm:

The proposed algorithm consists of the following steps:

1. Preparing raw images from dataset
2. Generating .mat files from the images
3. Generating ground values in csv files for train set
4. Resizing the images
5. Loading the train set and csv files
6. Building layers to form a network or model
7. Loading the model
8. Generating the density maps
10. Calculating the crowd count
11. Alerting by sending message to security after detecting dense crowd.

Implementation of Proposed Solution:

The implementation of this project consists of the following modules:

- **i.** Data Preparation
- **ii.** Model building
- **iii.** Crowd Counter
- **iv.** Alarming User

Data Preparation

In the data preparation the raw images are converted into .mat files using matlab. this .mat file contains two arrays , one array is of size N by 2 , the 2 is for the X and Y coordinate for targeted object pixels, and N is the total number of pixels . Finally it creates a directory where all .mat file of images are stored . It implements a guassian method to calculate the ground truth values of the train set images and stores in a csv file.

```
dataset = 'A';
dataset_name = ['shanghaitech_part_' dataset '_patches_' num2str(N)];
path = ['../data/original/shanghaitech/part_' dataset '_final/train_data/images/'];
output_path = '../data/formatted_trainval/';
train_path_img = strcat(output_path, dataset_name,'/train/');
train_path_den = strcat(output_path, dataset_name,'/train_den/');
val_path_img = strcat(output_path, dataset_name,'/val/');
val_path_den = strcat(output_path, dataset_name,'/val_den/');
gt_path = ['../data/original/shanghaitech/part_' dataset '_final/train_data/ground_truth/'];
```

Model Building:

For implementing the model, the images in the dataset and their data are preloaded into the CPU RAM so as to decrease the delay in the processing and stored in the data_files list which is used in further processing. The list of images, data obtained from previous step are resized to quarter of its width and height and reshaped as previous and is appended to list.

```
for idx in id_list:
    if self.pre_load:
        blob = self.blob_list[idx]
        blob['idx'] = idx
    else:
        fname = files[idx]
        img = cv2.imread(os.path.join(self.data_path,fname),0)
        img = img.astype(np.float32, copy=False)
        ht = img.shape[0]
        wd = img.shape[1]
        ht_1 = (ht/4)*4
        wd_1 = (wd/4)*4
        img = cv2.resize(img,(wd_1,ht_1))
        img = img.reshape((1,1,img.shape[0],img.shape[1]))
        den = pd.read_csv(os.path.join(self.gt_path,os.path.splitext(fname)[0]
            + '.csv'), sep=',',header=None).as_matrix()
        den = den.astype(np.float32, copy=False)
```

Then Convolutional layers and Fully Connected layers are constructed. Convolutional layer is designed using Conv2d of the neural network model of torch, Normalization layer is also added to normalize the weights. Relu activation is used to activate the model layers. Fully Connected Layer is designed using the linear module unit of neural

network model. An average pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region which reduces the number of connections to the following layers.

The model built is set to train and weights and biases are applied to increase accuracy of the model.

```
def set_trainable(model, requires_grad):
    for param in model.parameters():
        param.requires_grad = requires_grad
```

```
def weights_normal_init(model, dev=0.01):
    if isinstance(model, list):
        for m in model:
            weights_normal_init(m, dev)
    else:
        for m in model.modules():
            if isinstance(m, nn.Conv2d):
                #print torch.sum(m.weight)
                m.weight.data.normal_(0.0, dev)
                if m.bias is not None:
                    m.bias.data.fill_(0.0)
            elif isinstance(m, nn.Linear):
                m.weight.data.normal_(0.0, dev)
```

Finally, Evaluation of the network is done by calculating the Mean Squared Error(MSE) and Mean Absolute Error(MAE) of the model by passing test data. The MSE and MAE is calculated using following functions.

$$MAE = \frac{1}{N}\sum_{1}^{N}|z_i - \hat{z}_i|, \quad MSE = \sqrt{\frac{1}{N}\sum_{1}^{N}(z_i - \hat{z}_i)^2}$$

Crowd Counter

After Building the model, Crowd Counter module is used for detecting anomalous situation in the image.

```
data_loader = ImageDataLoader(data_path, gt_path, shuffle=False,
    gt_downsample=True, pre_load=True)

for blob in data_loader:
    im_data = blob['data']
    gt_data = blob['gt_density']
    density_map = net(im_data, gt_data)
    density_map = density_map.data.cpu().numpy()
    gt_count = np.sum(gt_data)
    et_count = np.sum(density_map)
    mae += abs(gt_count-et_count)
    mse += ((gt_count-et_count)*(gt_count-et_count))
    if vis:
        utils.display_results(im_data, gt_data, density_map)
    if save_output:
        utils.save_density_map(density_map, output_dir, 'output_'
            + blob['fname'].split('.')[0] + '.png')

mae = mae/data_loader.get_num_samples()
mse = np.sqrt(mse/data_loader.get_num_samples())
print '\nMAE: %0.2f, MSE: %0.2f' % (mae,mse)

f = open(file_results, 'w')
f.write('MAE: %0.2f, MSE: %0.2f' % (mae,mse))
f.close()
```

Alarming the user:

Using twilio, an alert message is sent to the respective ones. Account ID and authentication token are generated during the account creation in twilio and also number from which message has to be sent is given by twilio. We only need to pass phone number to whom message has to be sent and the body of the message.

```
from twilio.rest import Client

account_sid = 'ACc10f0bd86f3db6205a04dcf64
auth_token = '33a37df8c421d5f52fb041090043
client = Client(account_sid, auth_token)

message = client.messages \
    .create(
         body=msg,
         from_='+12029372090',
         to='+919951695278'
     )

print(message.sid)
```

RESULTS AND DISCUSSION

After running matlab code in the matlab it generates the ground truth values and mat files for each. It creates a separate folder to store all the mat files and the csv file for the given data. After processing all the images they stores in a folder , mat files contains two arrays , one array is of size N by 2 , the 2 is for the X and Y coordinate for targeted object pixels

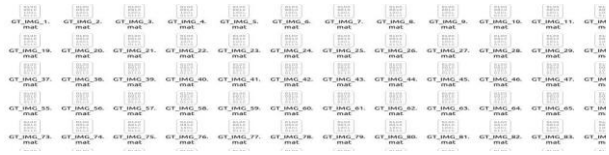

Figure 3. Folder output

The model building is done after the data preparation and MSE, MAE are printed .

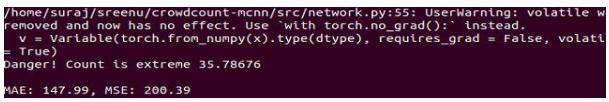

Figure 4. Built Model MSE, MAE

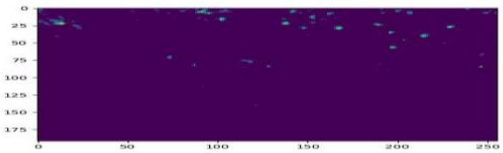

Figure 5 Density map of test image

When the input image is passed crowd count is generated and density maps are shown. The predicted count is transfered to the mobile using twilio API.

Figure 6. Alarm Message

CONCLUSION

Considering applications of the crowd detection, many attempts have been made to provide solution. However, existing models generally requires too much memory and power consumption in the computation, thus they can only be run on the server side with high GPUs. The proposed system provides an efficient solution using MCNN such that it is fast and accuracy and has less mean absolute error ,mean square error and also suitable for the small embedded systems. In the future work automated clustering techniques can be use to find extreme situations in the generated density maps and also increase accuracy by using other CNN methods

REFERENCES

[1] N. Dalal and B. Triggs, ‘‘Histograms of oriented gradients for human detection,’’ in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Jun. 2005, pp. 886–893.

[2] N. Paragios and V. Ramesh, ‘‘A MRF-based approach for real-time subway monitoring,’’ in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Dec. 2001, p. I.

[3] P. Sabzmeydani and G. Mori, ‘‘Detecting pedestrians by learning shapelet features,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2007, pp. 1–8.

[4] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, ‘‘Deep people counting in extremely dense crowds,’’ in Proc. 23rd ACM Int. Conf. Multimedia, Oct. 2015, pp. 1299–1302.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘‘Imagenet classification with deep convolutional neural networks,’’ in Proc. Adv. Neural Inf. Process. Syst., Jan. 2012, pp. 1097–1105.

[6]] M. Fu, P. Xu, X. Li, Q. Liu, M. Ye, and C. Zhu, ‘‘Fast crowd density estimation with convolutional neural networks,’’ Eng. Appl. Artif. Intell., vol. 43, pp. 81–88, Aug. 2015.

[7] P. Sermanet, S. Chintala, and Y. Lecun, ‘‘Convolutional neural networks applied to house numbers digit classification,’’ in Proc. 21st Int. Conf. Pattern Recognit., Nov. 2012, pp. 3288–3291

[8] A. Marana, L. D. Costa, R. A. Lotufo, and S. A. Velastin, ‘‘On the efficacy of texture analysis for crowd monitoring,’’ in Proc. Int. Symp. Comput. Graph. Image Process. Vis., Oct. 1998, pp. 354–361.

[9] C. Zhang, H. Li, X. Wang, and X. Yang, ‘‘Cross-scene crowd counting via deep convolutional neural networks,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2015, pp. 833–841

[10] E. Walach and L. Wolf, ‘‘Learning to count with cnn boosting,’’ in Proc. 14th Eur. Conf. Comput. Vis., Oct. 2016, pp. 660–676.

[11] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, ‘‘Single-image crowd counting via multi-column convolutional neural network,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2016, pp. 589–597.