

A COMPARISON OF MACHINE LEARNING METHODS FOR QUEUE LENGTH DETECTION

Mehmet Eren YEŞİLYURT¹, Mehmet Serdar GÜZEL² and Ebru AKÇAPINAR SEZER³

¹Computer Engineering, Kastamonu University, Kastamonu, TÜRKİYE

²Computer Engineering, Ankara University, Ankara, TÜRKİYE

³Computer Engineering, Hacettepe University, Ankara, TÜRKİYE

ABSTRACT. Queues are formed by people waiting for a service in public institutions and they can be defined as orderly groups of people. Automatically counting the number of people waiting in a queue through video camera footage would provide these institutions with valuable information with regards to customer service quality. In this paper, our goal is to compare several machine learning methods for finding the total number of people waiting in a queue given video camera frames. We approached this problem as a regression task. We used a subset of the Collective Activity Dataset and compared three different methods. The first two methods used bounding box coordinates and orientations provided by the dataset, while the last method utilized the bounding box coordinates to extract feature maps from the frames using RoiAlign. The first method used XGBoost, while the latter methods used Convolutional Neural Networks (CNNs). Results show that the method using RoiAlign presents the best prediction performance in terms of mean squared error and mean absolute error, compared to other methods.



1. INTRODUCTION



A queue is a group of people waiting to receive some service in an organized manner. Detecting the number of people waiting in a queue has important applications for retail stores and public institutions such as hospitals. Knowing the queue length density for different time periods would allow public-serving facilities to optimize their human resource allocation in order to reduce waiting times and improve their quality of service.

Compared to a crowd, a queue is characterized by its orderly structure. The people comprising a queue form straight or curved lines, and they usually face the same

Keywords: Queue length detection, crowd counting, machine learning.

 myesilyurt@kastamonu.edu.tr-Corresponding author;  0000-0002-7322-5572

 mguzel@ankara.edu.tr;  0000-0002-3408-0083

 ebru@hacettepe.edu.tr;  0000-0002-9287-2679.

general direction. The known literature mostly focuses on the crowd counting problem, which has a similar objective to the queue length detection problem. Crowd counting can be implemented through detection, regression or density map estimation [7] methods. Detection based methods are used to detect hand-crafted features such as the existence of body parts or body appearance [6] to find the total body count. Object detection techniques such as YOLO and Faster-RCNN have also been used for crowd counting [8, 9, 10, 11]. In contrast to object detection methods, regression based methods predict the total count of people in an image directly. Finally, the density map estimation methods work by creating a crowd density map which is then used to find the total count. Since regression and density map estimation methods do not rely on detecting and counting people individually, they are more resilient to occlusion problems.



FIGURE 1. A sample frame from the collective activity dataset [1].

ACTi, which is a corporation that provides video analytics solutions, offers a queue management system that can determine the amount of time a person has spent waiting on a queue [13]. The system works by counting the number of people in a region of interest determined by store managers [14].

Saini et al. used bounding boxes obtained from an object detector to estimate the number of people waiting in the queue in a given frame [15]. Their method assumes that a queue lies on a straight line on the image and fits a line in the form of $y=mx+b$

that minimizes the distance to the midpoints of the bounding boxes. To estimate the parameters of this line, they formulate the problem as an optimization problem and solve it using convex hull optimization.

In [16], Wu et al. model scenes as `actor-relation graphs` in order to classify collective and individual activities carried out by people in video streams. They use RoiAlign to extract features for each bounding box, which is then used to obtain feature vectors. These vectors are then used to build actor-relation graphs. After graph convolution and pooling, classifiers predict group activity and individual activity carried out by actors.

In this paper, we approached the queue length detection task as a regression problem. Given an image, our objective was to count the total number of people waiting in a queue while disregarding those that were not part of a queue. To this end, we evaluated 3 different methods on a subset of the collective activity dataset.

2. DATA AND METHODS

The Collective Activity dataset was created by Choi et al. with the goal of classifying collective activities carried out by groups of individuals [1]. The dataset is made up of 44 short videos. In each video, there is a small number of people carrying out a group activity such as talking, queuing or walking. Labels provided for each video contain the bounding box coordinates, individual activity and pose for every tenth frame.

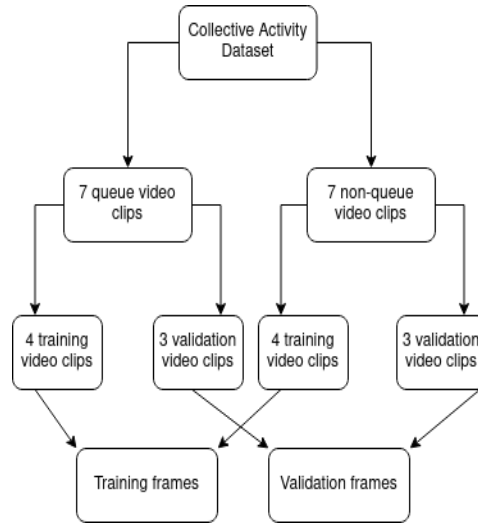


FIGURE 2. Dataset preparation steps.

To obtain the subset of the Collective Activity dataset used in this study, we used all 7 of the video clips that contained queues as well as 7 video clips that contained no queues. For both video sets, 4 out of 7 video clips were chosen for the training and the remaining 3 video clips were reserved for validation. Separating video clips in this manner ensured that the frames in the validation set and the frames in the training set came from different video clips. From each video clip in the training set, 21 frames were selected equidistantly. Similarly, 7 frames were extracted from every video clip in the validation set. This process resulted with 168 frames in the training set and 42 frames in the validation set with a validation/train split ratio of 20%. For each frame, the total number of individuals labeled as `queuing` served as the target variable of the regression task.

The first method we used was XGBoost utilizing tabular features. XGBoost, which stands for `Extreme Gradient Boosting`, was created by Chen and Guestring and it is defined as a `scalable tree boosting system` [2]. We used midpoint coordinates of each bounding box and pose information as features. Since each frame contains a variable number of individuals, and therefore, a variable number of bounding boxes; there are a different number of features for each frame. To get a fixed number of features to be given to the model, we assumed a limit of 14 people in a single frame. If the number of people in the frame was less than this value, the features of the individuals that were not present were set to 0. We used the same limit in the other methods as well. For the XGBoost model, we ended up with 42 features for each frame: 14 X coordinates, 14 Y coordinates and 14 pose labels.

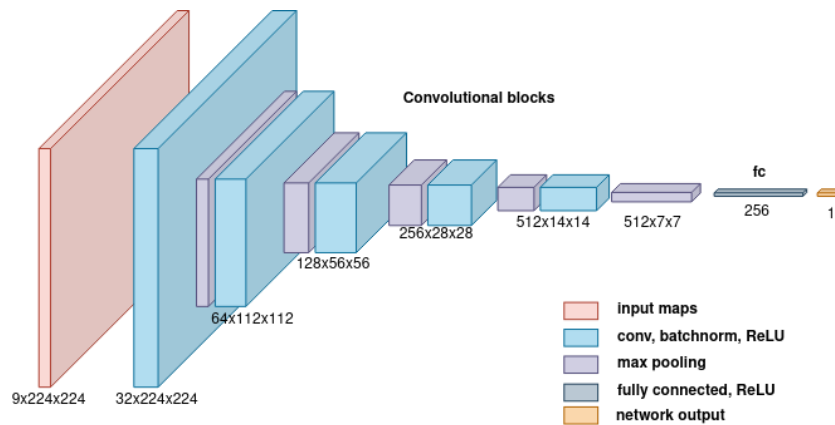


FIGURE 3. CNN model architecture.

The second method utilized a convolutional neural network (CNN). CNNs are deep learning models that are appropriate for input data that has regular spatial

structure, such as images [3]. The CNN architecture we used can be seen in Figure 3. Similar to VGG, the architecture is comprised of a number of convolutional layers with 3×3 kernels and max pooling, followed by a fully-connected layer. The input dimensions are $9 \times 224 \times 22$. The channels in the input correspond to different human orientations in the dataset. To create the input for the model, we obtained the middle point coordinates of each bounding box in a frame and scaled these coordinates to fit in a 224×224 grid. We then marked these coordinates in their respective pose channel. For example, if a person's rescaled bounding box coordinates were calculated as (100, 200) and that person's orientation was labeled as facing right, then the respective coordinate in the first channel (which corresponds to the orientation `right`) was set to 1. The persons with unspecified orientations were placed in the first channel.

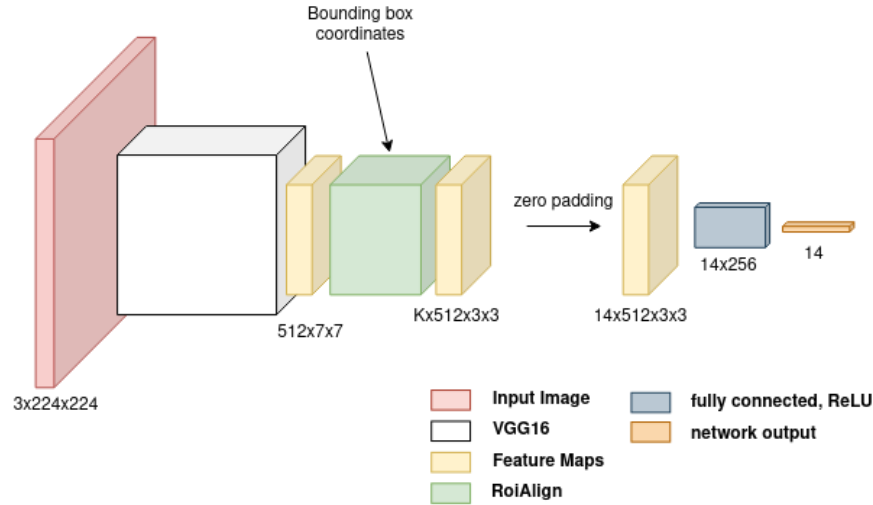


FIGURE 4. RoiAlign model architecture.

The third method also makes use of convolutional neural networks and its architecture can be seen in Figure 4. This network corresponds to the feature extractor stage in [16] by Wu et al. The model is made up of a feature extractor followed by a classifier that utilizes RoiAlign, which was introduced in [4] by He et al for extracting feature maps from a region of interest. The feature extractor takes a $3 \times 224 \times 224$ input image and creates feature maps of size $512 \times 7 \times 7$. We used a VGG16 [5] model without the fully-connected layers as the feature extractor. The classifier takes the output of the feature extractor and bounding box positions as input. Using RoiAlign, the classifier extracts 3×3 feature maps for each bounding box in the frame. This is followed by fully-connected layers. The model outputs a

value for each bounding box in the frame that represents the probability of the person in the bounding box belonging to a queue. We can then use the number of positive predictions output by the model as the number of people predicted to be in a queue for a given frame.

The XGBoost model was trained with default hyperparameters. To train the CNN and CNN RoiAlign models, we used the Adam [12] optimizer. After trying learning rates that ranged from 10^{-5} to 10^{-2} , we used a learning rate of 10^{-4} for the CNN model and a learning rate of 10^{-5} for the CNN RoiAlign model. Each training run lasted for 50 epochs, with early stopping after 10 epochs with no improvement.

TABLE 1. Evaluation results (Average of 5 training runs).

Method	Mean squared error	Mean absolute error
XGBoost	11.20	2.22
CNN	10.58	2.99
CNN RoiAlign	4.28	0.97

3. RESULTS

The results are shown on Table 1. We used mean squared error (MSE) and mean absolute error (MAE) to evaluate and compare each method. Due to the small size of the test set, there is a possibility of high variance in the obtained results. Because of this, we trained and evaluated each method 5 times and presented the average values obtained from these training runs. The network that utilized RoiAlign had the best MSE and MAE scores out of all methods. The CNN model that only used bounding box coordinates and pose information was very similar to the XGBoost model in terms of the MSE, while the XGBoost model gave better results in terms of the MAE.

According to the results shown in Table 1, it can be seen that the CNN RoiAlign method yields the best results compared to other methods, while the results for XGBoost and CNN in terms of the mean squared error were similar. The RoiAlign method makes use of extracted feature maps from the input image, while the XGBoost and the CNN models used only the bounding box position and pose information. The lower performance of the XGBoost and CNN models may have been caused by their direct dependency to bounding box coordinates provided in pixel coordinates. Since the dataset contained a mix of indoor and outdoor scenes with different locations and camera angles, the physical distances related to pixel distances were different for each scene.

4. CONCLUSION

This paper presents a comparison of different Machine and Deep Learning based methods for Queue Length Detection problem. In order to conduct experiments, a subset of the Collective Activity Dataset was employed. Three different methods were compared. While the first two methods, XGBoost and CNNs, used bounding box coordinates and orientations provided by the dataset; the final method, based on CNNs architecture, utilized the bounding box coordinates to extract feature maps from the frames using RoiAlign. Results showed the superiority of the final method over previous methods. It should be noted that the lower performance of the first two methods may have been caused by their direct dependency on pixel coordinates. Even though the CNN RoiAlign method still utilizes pixel coordinates of the bounding box locations, we found that using RoiAlign to extract feature maps for each bounding box and then classifying these boxes individually gives better results compared to other methods. Authors are planning to apply more complex Deep Learning architectures and larger datasets to the problem at hand in future work.

Author Contribution Statements Mehmet Eren Yeşilyurt analyzed the data, performed the computation work, prepared figures and/or tables. Mehmet Serdar Güzel conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work. Ebru Akçapınar Sezer analyzed the data, authored or reviewed drafts of the article, and approved the final draft.

Declaration of Competing Interests The authors declare that they have no competing interests.

Acknowledgement This study is supported by TUBITAK TEYDEB (Project Name: EYEIN: HASTANE İÇİ ORTAM İZLEME VE DURUM TESPİTİ İLE KALABALIK YÖNETİMİ, Project No: 3220818).

REFERENCES

- [1] Wongun, C., Shahid, K. and Savarese, S., What are they doing?: Collective activity classification using spatio-temporal relationship among people, *IEEE 12th Inter. Conf. on Comp. Vis. Wor. ICCV Work.*, (2009), 1282-1289, <https://doi.org/10.1109/ICCVW.2009.5457461>.
- [2] Chen, T. and Guestrin, C., XGBoost, *Proc. of the 22nd ACM SIGKDD Int. Conf. on Know. Disc. and Data Min.*, (2016).
- [3] Yamashita, R., Nishio, M., Do, R. K. G. et al., Convolutional neural networks: an overview and application in radiology, *Ins. Ima.*, 9 (2018), 611-629, <https://doi.org/>

- 10.1007/s13244-018-0639-9.
- [4] He, K., Gkioxari, G., Dollar, P. and Girshick, R., Mask R-CNN, *arXiv:1703.06870*, (2018), <https://doi.org/10.48550/arXiv.1703.06870>.
- [5] Simonyan, K. and Zisserman, A., Very deep convolutional networks for large-scale image recognition, *arXiv:1409.1556*, (2015), <https://doi.org/10.48550/arXiv.1409.1556>.
- [6] Khan, M. A., Menouar, H. and Hamila, R., Revisiting crowd counting: State-of-the-art, trends, and future perspectives, *Imag. and Vis. Comp.*, 129 (2023), 104597, <https://doi.org/10.1016/j.imavis.2022.104597>.
- [7] Sindagi, V. A. and Patel, V. M., A survey of recent advances in CNN-based single image crowd counting and density estimation, *Pat. Rec. Let.*, 107 (2018), 3-16, <https://doi.org/10.1016/j.patrec.2017.07.007>.
- [8] Ruchika, R., Purwar, K. and Verma, S., Analytical study of YOLO and its various versions in crowd counting, *Int. Data Com. Tech. and Int. of Thi.*, (2022), 975-989, https://doi.org/10.1007/978-981-16-7610-9_71.
- [9] Valencia, I. J. C., Dadios, E. P., Fillone, A. M., Puno, J. C. V., Baldovino, R. G. and Billones, R. K. C., Vision-based crowd counting and social distancing monitoring using tiny-YOLOv4 and DeepSORT, *IEEE Int. Sma. Cit. Conf. (ISC2)*, (2021), 1-7, <https://doi.org/10.1109/ISC253183.2021.9562868>.
- [10] Muzamal, J. H., Tariq, Z. and Khan, U. G., Crowd counting with respect to age and gender by using faster R-CNN based detection, *Int. Conf. on Appl. and Eng. Math. (ICAEM)*, (2019), 157-161, <https://doi.org/10.1109/ICAEM.2019.8853723>.
- [11] Akbar, N. and Jamal, E. C., Crowd counting using region convolutional neural networks, *8th Int. Conf. on Elec. Eng., Comp. Sci. and Info. (EECSI)*, (2021), 359-364, <https://doi.org/10.23919/EECSI53397.2021.9624288>.
- [12] Kingma, D. P. and Ba, J., Adam: A method for stochastic optimization, *arXiv:1412.6980*, (2017), <https://doi.org/10.48550/arXiv.1412.6980>.
- [13] People Queue Detection - ACTi Corporation. Available at: <https://www.acti.com/technologies/people-queue-detection>. [Accessed July 2023].
- [14] ACTi Corporation, Queue management whitepaper, ACTi Corporation, (2023). Available at: <https://download.acti.com/?id=10618>.
- [15] ShivamJalotra, Queue-Detection, (2023). Available at: <https://github.com/jalotra/Queue-Detection>. [Accessed July 2023].
- [16] Wu, J., Wang, L., Wang, L., Guo, J. and Wu, G., Learning actor relation graphs for group activity recognition, *arXiv:1904.10117*, (2019), <https://doi.org/10.48550/arXiv.1904.10117>.