# Crowd behavior Analysis Using 3d Convolutional Neural Network

Divya R. Pillai
MTech student, Communication Engineering, divyarpillai265@gmail.com

Prof. Nandakumar P.
Professor in Electronics and Communication Department,
NSS College of Engineering, Palakkad

**Abstract**— A method called 3D Convolutional Neural Network is proposed for identifying crowd behaviours in visual scenes. The algorithm used is Back Propagation. By the technique the crowd scenes are divided into multiple consecutive frames, which forms the input. Convolution, sub sampling are performed to extract the crowd pattern. Neural network is trained to classify the extracted information i.e. it classifies the crowd behaviour.

**Index Terms**— Convolution, 3D Convolutional neural network, Crowd behavior, Regression, Sobel Filtering .

## 1 INTRODUCTION

Video surveillance has become a topic more and more significant with the advent of technology and because of the increasing need for security. One particular class of public security issues is that involving a mass of people gathering together (crowding), such as public assemblies, sport competitions, demonstrations (eg., strikes, protests), airports, railway stations etc. Because of the high level of degeneration risk, the security of public events involving large crowd has always been of high concern to relevant authorities. Especially, this problem has started to draw attention of the research community for automatic detection of abnormal crowd behaviours during public events.

Technically speaking, crowd behaviour analysis can be divided into two tasks: (1) motion information extraction and (2) classification of crowd behaviour. Many techniques have been tried, but a most innovative technique called 3D Convolutional Neural Network [1] is tried to analyse the crowd behaviour. This technique has been successfully implemented for the human action recognition [1].

Many works on crowd density estimation, crowd tracking and crowd motion and behavions have been reported. [2] illustrates how the motion pattrern of humans was utilised to better identify pedestrains in a scene. Detection using several cameras [3] overcome the problem of occlusion and low resolution cameras. Ma *et al*. [4] computed the crowd density by taking geometrical distribution from ground plane to image plane. Other methods rate the crowd density for an example, Marana *et al.* [5] computed GLDM (Gray Level Dependence Matrix). Optical flow [6] method finds the similarities in the neighborhood of a block and estimate the matching of intensity levels across the frames. BraMBLe [7] (Bayesian Multiple Blob Tracker) tracks people across the frames and sum up this information to compute multiple people velocity.

Crowd event modelling was proposed by Andrade *et al*. [8] and it was experimented in a constrained environment. HMM was used to model crowd motion based on optical flow. KLT (Kanade-Lucas-Tomasi) tracker [9][10] helps to combine feature point extraction and tracking. A SRE (Scenario Recognition Engine) [11] was initially used for indoor event detection. But with additional facility functions provided within the SRE [12], it can cope with crowd together with crowd attributes.

As deep learning models, Convolutional Neural Network (CNN) architecture have been applied on 2-D images and they have yielded very competitive performance in many image processing tasks, but their application in video stream classification is still an open and unexplored area. In 3D convolution in the convolutional layers of CNNs so that discriminative features along both spatial and temporal dimensions. To understand convolutional neural network, a basic knowledge of neural networks is needed.

Neural network is a network or circuit of biological neurons. Connections of neurons are called weights and they are in general 2 types called positive weights (excitory connections) and negative weights (inhibitory connections ). Main parts of a neuron are cell body, axon, dendrites [13]. Connections formed between axon and dendrites are called synapses.

### 1.1 Problem Formulation

Videos of crowd scenes deliver challenging problems in computer vision. High object-densities in real world situations make individual object recognition and tracking impractical. So inorder to get a clear idea of the crowd behaviours with-

out understanding the actions of individuals, is often advantageous. Automated detection of crowd behaviours has numerous applications, such as crowd management, public space design, virtual environment, visual surveillance, intelligent environment.

## 2 3D Convolutional neural Network

### 2.1 Theory

My project is being implemented using the technique 3D Convolutional Nueral Network. But before dealing with the technique let we go through some basics. In this technique the main principle used often is "Convolution". Convolution is a mathematical operation on two functions m and n, generating a third function that is typically viewed as a modified version of one of the original functions, giving the overlapping area between the two functions as a function of the amount that one of the original functions is translated. The convolution of m and n is written as m*n, using an asterisk or star. It is defined as the integral of the product of the two functions after one is reversed and shifted. As such, it is a particular kind of integral transform:

$$(m*n)(t) = \int_{-\infty}^{\infty} m(\tau)n(t-\tau)dT$$

While the symbol *t* is used above, it need not represent the time domain.

### 2.2 Convolutional Neural Network (2D)

Convolutional neural networks are also known as "shared weight" neural networks. ConvNets are the adaptation of multilayered neural deep architectures to deal with real world data. This is done by the use of local receptive fields (better known as kernels) whose parameters are forced to be identical for all its possible locations of input array, a principle called weight sharing. The idea is that a small kernel window is moved over each node from a prior layer. In the CNN architecture, the sharing of weights over processing units reduces the number of free variables, increasing the generalization performance of the network. Weights are replicated over the input image, leading to intrinsic insensitivity to translations in the input.

A typical convolution framework is shown in Fig.1. Multiple planes are usually used in each layer (called Features maps (FMs)) so that multiple features can be detected. These layers are called convolutional layers. The network is trained with the usual backpropagation gradient-descent procedure. 2-D CNNs are applied on image dataset to classify them and extract spatio features.

### 2.3 3D Convolutional Neural Network

In 2D CNNs, features are computed from the spatial dimensions by applying convolution on 2D feature maps. But for action recognition in videos along with spatial features, motion information encoded in multiple contiguous frames is also captured. To receive the motion information in video analysis, it is proposed to perform 3D convolution in the convolutional layers of CNNs so that discriminative features along both spatial and temporal dimensions are captured. 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together. Fig 1 shows how the kernals are applied in 3D convolution. By this construction, the feature-maps in the convolution layer are connected to multiple contiguous frames in the previous layer, thereby capturing motion information.

A 3D convolutional kernel weights are replicated across the entire cube so it can only extract one type of features from the frame. An important design principle of CNNs is that the number of feature maps should be increased in late layers by generating multiple types of features from the same set of lower-level feature maps. Similar to the case of 2D convolution, this can be achieved by applying multiple 3D convolutions with distinct kernels to the same location in the previous layer.


Fig. 1. In 3D convolution, to the overlapping 3D cubes in the input video same 3D kernals is applied to extract motion features.

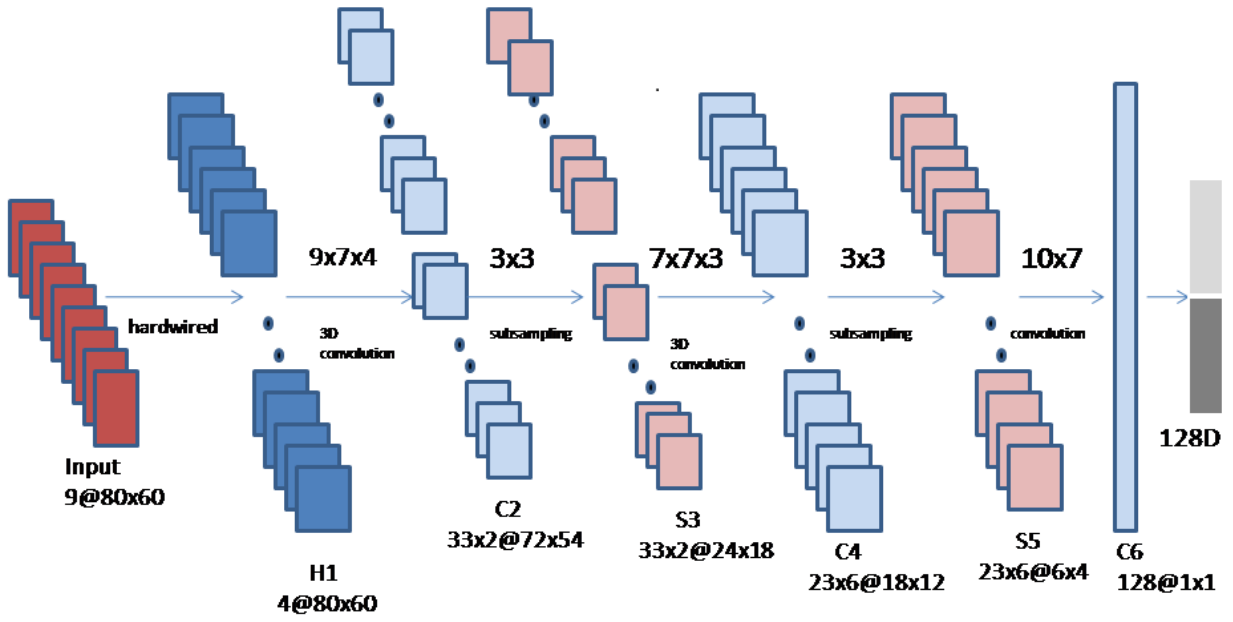



Fig 2. A 3D CNN architecture for crowd behavior recognition. This includes 1 hardwired layer, 3 convolution layers and 2 subsampling layers

## 2.4 Architecture

Based on the 3D convolution described above, a variety of CNN architectures can be devised. In the following, we describe a 3D CNN architecture that we have developed for crowd behavior recognition. In this architecture shown in Fig 2., we consider 7 frames of size 80×60 centered on the current frame as inputs to the 3D CNN model. We first apply a set of hardwired kernels to generate multiple channels of information from the input frames. This results in 43 feature maps in the second layer in 5 different channels known as gray, gradient-x, gradient-y, optflow-x, and optflow-y. The gray channel contains the gray pixel values of the.

7 input frames. The feature maps in the gradient-x and gradient-y channels are obtained by computing gradients along the horizontal and vertical directions, respectively. 7 input frames, and the optflow-x and optflow-y channels con- tain the optical flow fields, along the horizontal and vertical directions, respectively, computed from adjacent input frames. This hardwired layer is used to encode our prior knowledge on features, and this scheme usually leads to better performance as compared to random initialization.

We then apply 3D convolutions with a kernel size of 9 × 7 × 3 (9 × 7 in the spatial dimension and 3 in the temporal dimension) on each of the 5 channels separately. To increase the number of feature maps, two sets of different convolutions are applied at each location, resulting in 2 sets of feature maps in the C2 layer each consisting of 33 feature maps. In the subsequent subsampling layer S3, we apply 3 × 3 subsampling on each of the feature maps in the C2 layer, which leads to the same number of feature maps with reduced spatial resolution. The next convolution layer C4 is obtained by applying 3D convolution with a kernel size of 7 × 7 × 3 on each of the 5 channels in the two sets of feature maps separately. To increase the number of feature maps, we apply 3 convolutions with different kernels at each location, leading to 6 distinct sets of feature maps in the C4 layer each containing 23 feature maps. The next layer S5 is obtained by applying 3×3 subsampling on each feature maps in the C4 layer, which leads to the same number of feature maps with reduced spatial resolution.

At this stage, the size of the temporal dimension is already relatively small (3 for gray, gradient-x, gradient-y and 2 for optflow-x and optflow-y), so we perform convolution only in the spatial dimension at this layer. The size of the convolution kernel used is 10 × 7 so that the sizes of the output feature maps are reduced to 1×1. The C6 layer consists of 128 feature maps of size 1 × 1. By the multiple layers of convolution and subsampling, the 9 input frames have been converted into a 128D feature vector capturing the motion information in the input frames. The output layer consists of the same number of units as the number of actions, and each unit is fully connected to each of the 128 units in the C6 layer.

## 2.4 Back propagation Algorithm

It is the most simplest and general methods used for supervised training of multilayered neural network. Back propagation works by approximating the non-linear relationship be-

tween input and output by adjusting the weights internally. Back propagation networks have 2 stages, training and testing. Training is giving an idea to the network about the input – target pairs and testing is checking the output so that it matches with the target to which the network is trained. Fig 3. explains the algorithm steps clearly.

Step1. Apply the inputs to the network and work out the output – note that initial output could be random, as the initial weights were random numbers.

Step2. Compute the error for neuron Y. The error is What you want – What you actually get, in other words:
ErrorY = OutputY (1-OutputY)(TargetY – OutputY)
Due to the Sigmoid Function, the "Output(1-Output)" term is necessary.

Step3. Change the weight. Let W+XY be the new (trained) weight and WXY be the initial weight.
W+XY = WXY + (ErrorY x OutputX)
Note that it is the output of the connecting neuron (neuron X) we use (not Y). We update all the weights in the output layer in this way.

Step4. Compute the Errors for the hidden layer neurons. Unlike the output layer we can't calculate these directly (because we don't have a Target), so we Back Propagate them from the output layer (hence the name of the algorithm). This is done by taking the Errors from the output neurons and passing them back through the weights to get the hidden layer errors. For example if neuron X is connected as shown to Y and Z then we take the errors from Y and Z to generate an error for X.
ErrorX = Output X (1 - Output X)(ErrorY WXY + ErrorZ WXZ)
Again, the factor "Output (1 - Output )" is present because of the sigmoid squashing function.

Step5. Having obtained the Error for the hidden layer neurons now proceed as in step 3 to change the hidden layer weights. By repeating this method we can train a network of any number of layers, until the error is minimum.

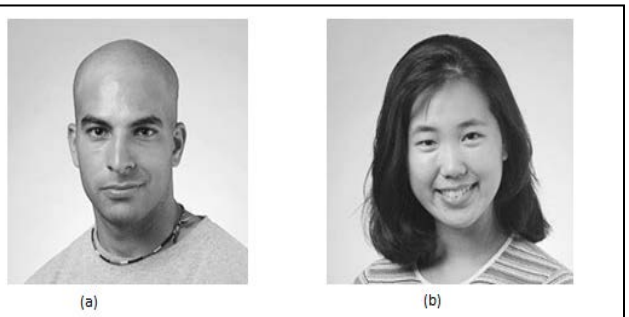



Fig 4. Shows the dataset. (a) is the input image that trains the network (b) is the target image that tests the network.

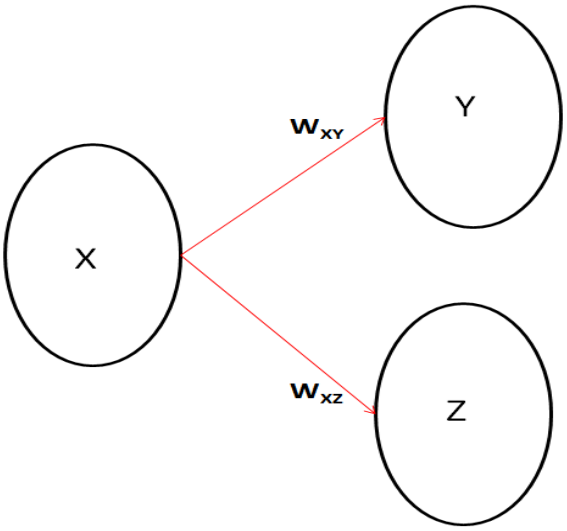



Fig 3. A connection to explain the algorithm.W stands for the weight. X is the input and Y and Z are the outputs.

## 2.6 Modeling Neural Network using Backpropagation

We give a training set which train the network and sets the weights with minimum error.A testing set is the actual input and the target is the desired output to which the obtained output is compared. Here we have modelled a neural network using an input image and a target image. Then the network is trained using these images. We can study the performance, training state and regression from from the graphs provided from the neural network modeled. Fig 5, Fig 6 and Fig 7 shows the performance graph, training graph and regression graph respectively when this input image and the target image is convolved.

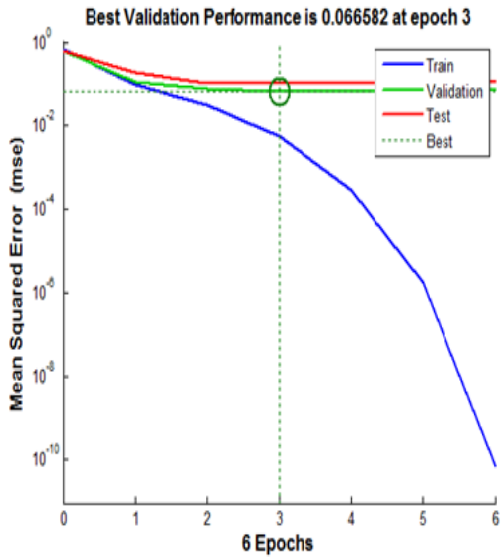
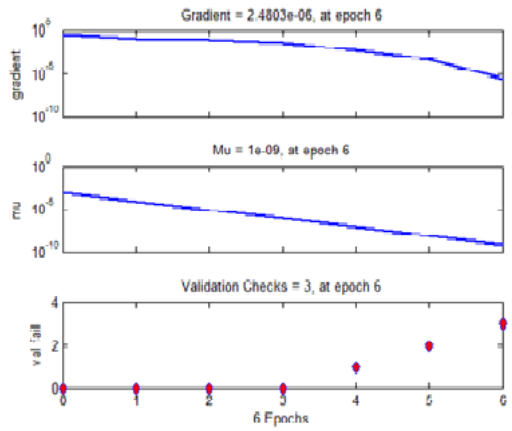



Fig 4. Shows the dataset. (a) is the input image that trains the network (b) is the target image that tests the network.




. Fig 6. Training graph when 2 2D images are convolved.

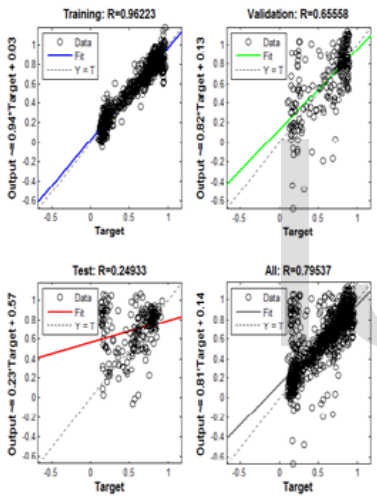



Fig 7. Regression graph when 2 2D images are convolved.


Fig 8. Training dataset. (a)and (c) shows the blocking crowd video shots, (b) and (d) shows the lane crowd video shots.

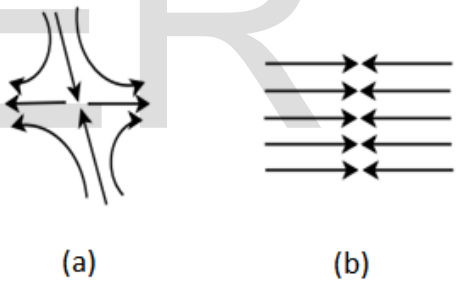



Fig 9. Crowd behaviors. (a) blocking (b) lane

## 3 BRIEF EXPLAINATION OF THE PROJECT

The crowd behaviour analysis is implemented using 3D convolutional neural network, because it has many advantageousness's, such as these networks can be used to extract patterns and detect trends that are too complex to be noticed by either humans or computer techniques and these networks can be trained and tested as well.

The input data is a video of few minutes. From the video, initially the crowd information is extracted and finally classifies the information to different categories. Crowd information extraction is done by the convolution layers and the classification is done by the neuron layers.

The experiment has mainly 3 steps; creating the database, training the neural network and testing the network. Here the training database contain 4 crowd videos; 2 blocking videos and 2 lane videos. Aim of the first step i.e. creating the database is to read the video files and finding 3D cnn features.

The final step is testing the network. Aim of this step is crowd behavior detection. The trained cnn feature initialized, neural network is created and the network is trained. Tested cnn feature values are compared and the crowd behavior is recognized i.e. either blocking or lane.

### 3.1 Crowd behaviors

In this work the recognized behaviors are blocking andd lane

[14]. Fig 9. gives an idea of the flow paths of blocking crowd and lane crowd.

**Blocking.** It is the characteristic behavior of mass in densely populated areas where the sorrounding mass prevents the desired motion of many people i.e. pedestrains moving in opposite direction block each other as density increases.

**Lane.** The flow paths are formed in opposite direction such that people moving against the flow will step aside to join the mass that is moviing in the same direction, hence avoiding any collisions.

## 4 RESULTS AND CONCLUSIONS

With the training dataset the neural network is trained and the features are extracted. The extracted features are saved along with the behavior. Then the network is tested by loading the video. This testing procedure is actually the crowd behavior detection. Features extracted in this step is compared to detect the behavior. Hence the behavior is recognised and the input video is played again with a text in it which shows the recognition i.e. letter B for blocking behavior and letter L for lane behavior. Fig 10 shows the video shots with the text in it implying that the crowd behavior is recognized.


Fig 10. Shows the recognized video shots. (a) and (c) are the recognized video shots of blocking crowd, (b) and (d) are the recognized video shots of lane crowd .

## ACKNOWLEDGMENT

The author would like to thank Prof. Nandakumar P for calling his attention to the paper , and an anonymous reviewer for helpful comments that improved the clarity of the paper.

## REFERENCES

[1] S. Ji, W. Xu, M. Yang, and K. Yu. " 3D convolutional neural networks for human action Recognition ". In ICML, 3362, 3366 [2010].

[2] P. Viola, M.J. Jones, D. Snow, Detecting Pedestrians Using Patterns of Motion and Appearance, International Journal of Computer Vision, 63(2), 2005, 153-161.

[3] Khan, S.M., Shah, M.: A multiview approach to tracking people in crowded scenes using a planar homography constraint. In: 9th European Conference on Computer Vision, LNCS, vol. 3954, pp. 133146. Springer, Heidelberg, 2006.

[4] Ruihua Ma, Liyuan Li, Weimin Huang, Qi Tian. On Pixel Count Based Crowd Density Estimation. IEEE Conference on Cybernetics and Intelligent Systems. December 1-3, 2004, Singapore.

[5] A.N. Marana, S.A. Velastin, L.F. Costa, R.A.Lotufo, Automatic Estimation of Crowd Density Using Texture, Safety Science, 28(3), 1998, 165-175.

[6] A.C. Davies, J.H. Yin and S.A. Velastin, Crowd Monitoring Using Image Processing, IEE Electronic and Communications Engineering Journal, 7(1), 1995, 37-47.

[7] M. Isard and J. MacCormick, BraMBLe: A Bayesian Multiple-Blob Tracker, Proc. International Conference on Computer Vision, Vancouver, Canada, 2001, vol.2 34-41.

[8] E. L. Andrade, S. Blunsden, R. B. Fisher. Modelling Crowd Scenes for Event Detection, in Proc. Int. Conf. on Pat. Recog. (ICPR'06), Vol 1, pp 175 - 178, HongKong, Aug. 2006.

[9] B.D. Lucas, T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision, International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981, 674-679.

[10] J. Shi, C. Tomasi, Good Features to Track, IEEE Conference on Computer Vision and Pattern Recognition, Seattle, USA, 1994, 593-600.

[11] T. Vu, F. Br´emond and M. Thonnat. Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition. The Eighteenth International JointConference on Artificial Intelligence (IJCAI'03), Acapulco, Mexico, 9-15 August 2003

[12] T. Vu. Scenario Description Language for Video Interpretation. Internal document. INRIA Sophia Antipolis, Revision 2007.

[13]http://www.mind.ilstu.edu/curriculum/neurons_intro/neurons_intro.php.

[14] Berkan Solmaz, Brian E. Moore, Mubarak Shah,"Identifying Behaviors in Crowded Scenes Using Stability Analysis for Dynamical Systems", *IEEE Trans. on Pattern Analysis and Machine Intelligence, v*ol. 34 no. 10 Oct.2012