Project Presentation
(Final - ESA)

Project Title : Stampede detection using MCNN

Project Team : Hritvik Patel PES1201700125
Shreyas BS PES1201700956
Archana P PES1201701543

Project Abstract and Scope

Why we have chosen this problem:

In a highly populated country like India, stampedes pose a huge threat to human lives. Our aim was to apply what we've learnt to solve a social problem. The advancements in hardware technology offer great potential to this model that can be used to predict a stampede and alert the concerned authorities. The idea of using basic math to save lives amused us.

Comparisons of our solution with other existing solutions:

Many of the existing solutions for stampede detection with images use image processing techniques for head counting. The classification would be solely based on the count of people. As in our approach, using a model would mean more learning than just the count of people and hence could offer more accurate classifications.

What is unique about our solution:

1. Our CNN uses filters of different sizes, which makes it scale invariant. Images used for training were clicked from different distances resulting in people appearing in various sizes. This approach helped our model perform better.
2. We also used an auto-encoder in order to achieve image compression without information loss, the output of which was fed into our Multi-column CNN.

Solution Architecture

The dataset was compiled from various sources and manually labelled. They were then resized into dimensions of 100x100 and fed into an auto-encoder as depicted in Fig.1.

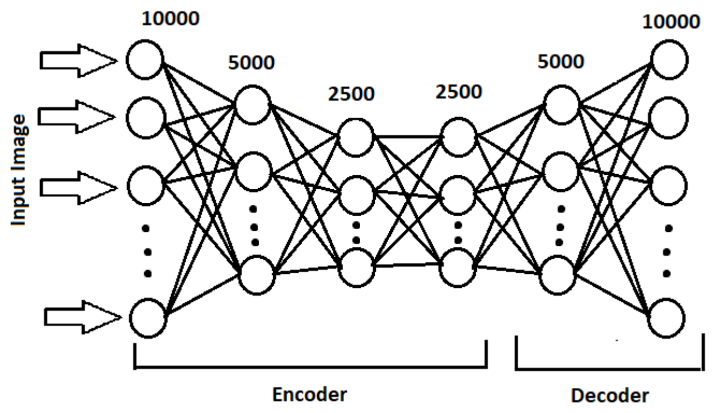


Fig.1

The auto-encoder was used to compress the dataset to a size of 2500 pixels(flattened).

The key layers of the auto-encoder are as described:

1. Dense layer 1: 10000 neurons (representing the flattened input image)
2. Dense layer 4: 2500 neurons (representing the final compressed image)
3. Dense layer 6: 10000 neurons (representing the reconstructed image after decompression)

The output obtained from layer 3(compressed), was then fed into a multi-column convolutional neural network. The auto-encoder was used to reduce the computational load on the MCNN.
The MCNN consisted of 3 columns, employing filters of various sizes, thereby achieving the scale invariance, which was necessary to classify the dataset having images of humans captured at different scales. The merged output if these columns was then fed through a network of dense layers, and finally to an output neuron with sigmoid activation. The output of this neuron was thresholded to give us the final classification. The architecture of the MCNN is depicted in Fig.2

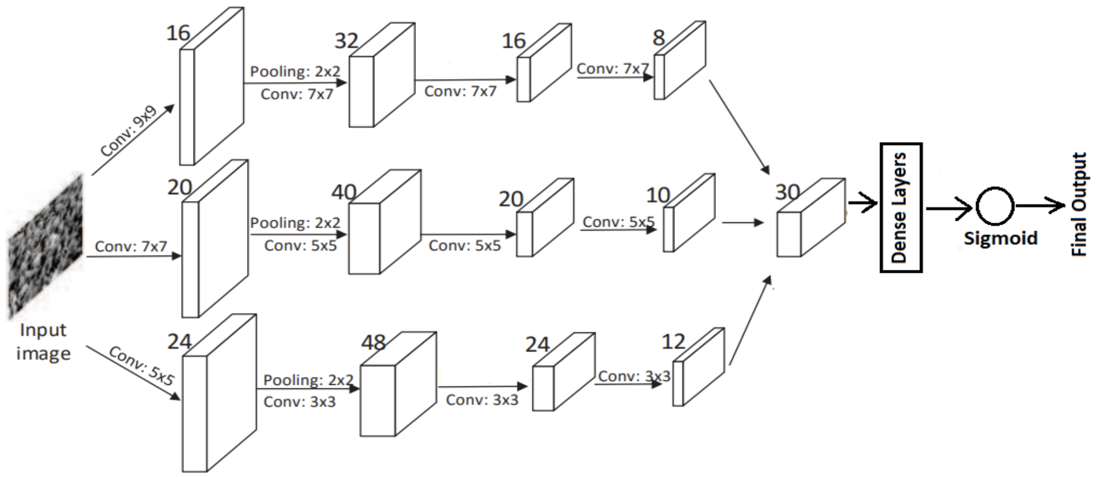


Fig.2

The configuration of the columns is briefly described.

Column 1:

Number of feature maps: 8

Size of feature map: 25 x 25

Column 2:

Number of feature maps: 10

Size of feature map: 25 x 25

Column 3:

Number of feature maps:12

Size of feature map: 25 x 25

These sets of feature maps were concatenated(merged) to give us a final set of 30 feature maps of size 25 x 25. The network of dense layers into which these activation maps were fed is summarized.

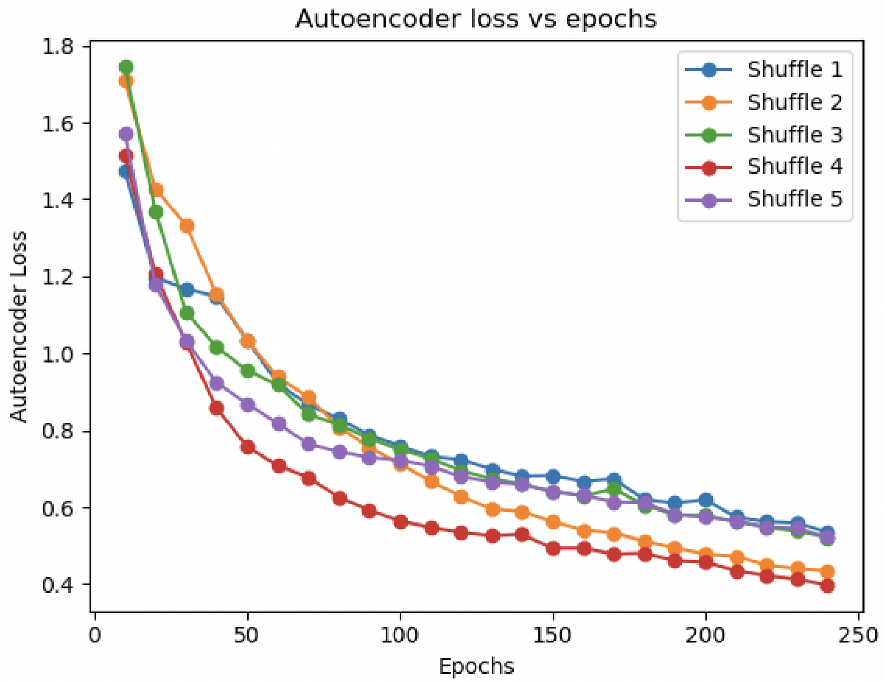
1. First dense layer: 18750 neurons
2. Second dense layer: 1875 neurons
3. Third dense layer: 187 neurons

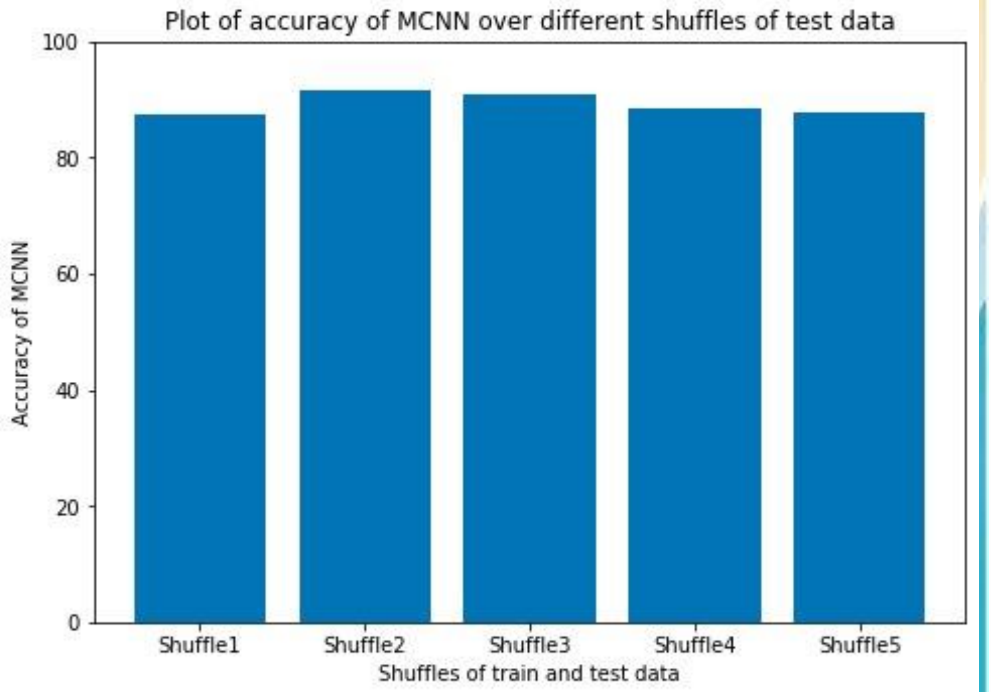
Solution

The compiled dataset had to be converted to grayscale and resized into dimensions of 100 x 100. These images were then fed into the auto-encoder. Adam optimiser was used to achieve faster convergence. The network was improved by using MSLE (Mean Squared Logarithmic Error) as the loss function. This was preferred over using MSE (Mean Squared Error) as the range of the pixel values in an image(target values) was high, and penalising large errors more than small ones had to be avoided.
The trained auto-encoder gave a loss of 0.47, when averaged over 5 cycles of dataset shuffling and training for 250 epochs.

The output of the encoder was manually reshaped into dimensions of 50 x 50 x 1 and was fed into the MCNN as input.

The MCNN was trained over the Adam optimiser, by using sigmoid cross entropy as the loss function. The network was trained for 20 epochs. The average accuracy of the MCNN over 5 shuffles of dataset was computed to be 89.24%.
The results obtained are visualised in the form of graphs.


Autoencoder loss vs epochs
Autoencoder Loss
Epochs
Shuffle 1
Shuffle 2
Shuffle 3
Shuffle 4
Shuffle 5
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
50
100
150
200
250


Plot of accuracy of MCNN over different shuffles of test data
Accuracy of MCNN
Shuffles of train and test data
Shuffle1
Shuffle2
Shuffle3
Shuffle4
Shuffle5
100
80
60
40
20
0

Constraints, Assumptions & Dependencies

Constraints

Our dataset was compiled from various sources and was not readily available. We had to label our images according to the class as stampede/non-stampede. We could find a limited amount of images which could be used for our application.

Assumptions

We had to label our images as stampede/non-stampede as there was no dataset with the label readily available. But we are sure that given a valid dataset, our model would still work.

Dependencies

Our model would require integration with hardware so that it could be fed with live images.

Future work plan

1. Usage of a larger and a valid dataset.

2. Possibly write an interface to feed in images captured during surveillance directly into the model.

References

1. https://zpascal.net/cvpr2016/Zhang_Single-Image_Crowd_Counting_CVPR_2016_paper.pdf
2. https://www.kaggle.com/danaelisanicolas/high-density- crowd-counting/data
3. https://bgfons.com/uploads/crowd/?C=D;O=A
4. https://towardsdatascience.com/guide-to-coding-a- custom-convolutional-neural-network-in-tensorflow-bec694e36ad3
5. https://towardsdatascience.com/deep-autoencoders-using-tensorflow-c68f075fd1a3

Thank You