TABLE OF CONTENTS

1. Introduction

Crowd counting and analysis is a topic that has gained a lot of popularity over the past few years because of its diverse applications such as crowd management, public space design etc. For example, at the platform of a railway station, global count maybe sufficient. At a mall however, crowd density mapping is also necessary since it evaluates which shop is more popular. The recent pandemic COVID-19 has proved the need for such crowd analysis techniques as widespread infection occurred through the same and it was crucial for us to develop a measure of the number of people that may congregate together in areas prone to overcrowding.

In a country like India, with a staggering population of over 135 crores and a population density of 464 people per km^2, it is practically impossible to resort to manual methods of counting. Hence, we need a fast, real-time, accurate and modular way to actively deploy a machine learning algorithm in the real world. In order for the model we have built to work in real time without the use of cloud computing, we need to deploy it as an edge-AI application. To this end, the NVIDIA Jetson Nano was chosen for the required embedded applications to be performed.

The approach taken towards achieving the main goal of the project was initially to perform feature extraction and obtain a regression model. This model takes interdependent local features from local spatial regions to be the input and people count from the individual regions as output. However, the results obtained on MATLAB software suggested that the trained model was overfit and hence our focus was shifted towards using a Convolutional Neural Network (CNN), which works better in real time as well.

1.1 Problem Statement

“Real Time, accurate Edge-application of Machine Learning Algorithms to count the number of people in a video feed in an area”

Crowd counting is a task that involves estimating the number of people in images or surveillance videos, and has garnered a lot of attention in the computer vision community due to its many applications, especially in security-related scenarios. Procedurally, a solution to crowd counting takes an input like an image or say a video clip. The output is a predicted number indicating the

number of people in the input. This has many challenges because of problems like large variations of density, scale, perspective, and severe occlusion in the captured video data.

Crowd analysis is a typical regression problem which resolves to approximate the crowd size by analyzing and exploiting various features of an image. In the surveyed literature, many features based on several attributes have been discovered and designed. In more recent times, some Deep Learning based counting methods have shown great promise and potential. Our project aims at creating a real time, accurate application of machine learning algorithms to count the number of people from the obtained video feed through a webcam in an area.

1.2 Brief Literature Survey

The literature review conducted for our project mainly revolved around papers whose key theme was training data using regression based/ CNN based models. Counting by regression is computationally the most efficient in that it creates a direct mapping between low level features and people count. In the regression papers that we studied, features were extracted from the frames of a video feed and fed to a pre-trained model that generated a numeric output. These methods, however, are outdated; the best and most efficient techniques now are those involving CNN and Deep Learning. In Ahmed F. Gad et al. 2017 [1], crowd density estimation avoids localizing individuals and depends on feature extraction which can be classified based on whether the features are extracted holistically or locally. Local feature extraction segments the image into segmented regions (SR) and the features are extracted from each of the regions. This method has two stages: feature selection and regression modeling. The results showed that it overcomes the linearity problem, it is robust and gives results of high accuracy.

CNN algorithm methods focus only on the local appearance features of crowd scenes and ignore the large-range pixel-wise contextual and crowd attention information. To overcome this problem, Spatial-/Channel-wise Attention Models are introduced into the Regression based CNN to approximate the density map, and is called SCAR [2]. In one such paper, Mohib Ullah et al. 2018 [3], a spatio-temporal deep CNN that exploits temporal information for spatial segmentation was proposed. It can be used as a pre-trained network. Its architecture comprises the input to the network being a set of three frames at the time instant

t−1, t, and t + 1. The network thereby learned visual and temporal features and utilized it for spatial segmentation. Hence, the output found was a binary mask which showed the region belonging to a pedestrian in the middle frame. In essence, the final output will be a frame with bounding boxes drawn around each individual person.

2. Methodology and Implementation

In order to test the regression model, MATLAB was used. The Mall Dataset was used as the primary dataset for training and testing. MATLAB code was written to extract features from the images and several models were trained using this, and the accuracies were compared. Regression Learner App in MATLAB was used to do this. For the CNN model, the pre-trained networks Faster RCNN and PedNet were implemented on Google Colab in order to avail the free GPU offered by Google.

2.1 Components Selection and Software platform

In order to test the accuracy and obtain results on the model we trained using the Mall Dataset, the project was implemented using the components as stated below:

Sl. No.	Components	Make	Specific Advantage
1.	MATLAB, version 2019b	MathWorks	For image processing and the Regression Learner App.
2.	Google Colab	Google	It is a python coding platform with hardware acceleration.
3.	Jeston Nano	NVIDIA	Ideal for Edge-AI applications
4.	Webcam	Logitech(C930 e model)	For image acquisition

MATLAB is a high-level language that we chose for our project, particularly to train, assess and export the regression model using the Regression Learner App. Google Colab is a platform that allows anyone to write and execute python code through the browser. It is particularly well suited for machine learning applications. Its Graphical Processing Unit on the cloud can be used to speed up time-consuming training phases.

2.2 Approach for Implementation

The following features were extracted from the images obtained from the Mall Dataset: Density, number of Edge-count Pixels, Minkowski Dimension, Gray Co-occurrence Matrix (determinant, trace and first Eigen value of this matrix were used as features). The Mall Dataset was split 70:30 as the number of images used for test: train respectively.

The new Mall Dataset was captured using a surveillance camera that was accessible publically in a shopping mall. It explores more challenging lighting conditions and glass surface reflections. The Mall dataset also covers more diversified crowd densities, from scattered to crowded, in addition to several activity patterns such as static and on-the-move crowds under a wide range of illumination environments at different points and times during the day. Additionally, as compared to the UCSD dataset, the Mall Dataset is subject to more harsh perspective distortions, which in turn causes larger variations in size and appearance of objects with respect to different depths of the scene. It also has more recurrent occlusion issues which are due to the objects present in the scene, for example- stalls, indoor plants that are present along the walking path. This dataset therefore offers a robust and challenging testing arena for a Crowd-counting model.

It is important to utilize effectively the correlation between features across regions. It is also necessary to bring about the sharing of information across regions to achieve more robust crowd counting. Without information sharing, the local measures are too noisy and brittle to be relied upon in isolation for estimating density.

All of the above processes were tested on MATLAB. However, when tested in the real world on images, the model did not perform as well as we expected it to, in that it was overfit. Hence, we implemented the Python code for CNN development on Google Colab. Faster RCNN and PedNet were tested. These modules detected the number of people in an image, then a snippet of code was written to count and display the number of detected people in the top-left corner of the image.

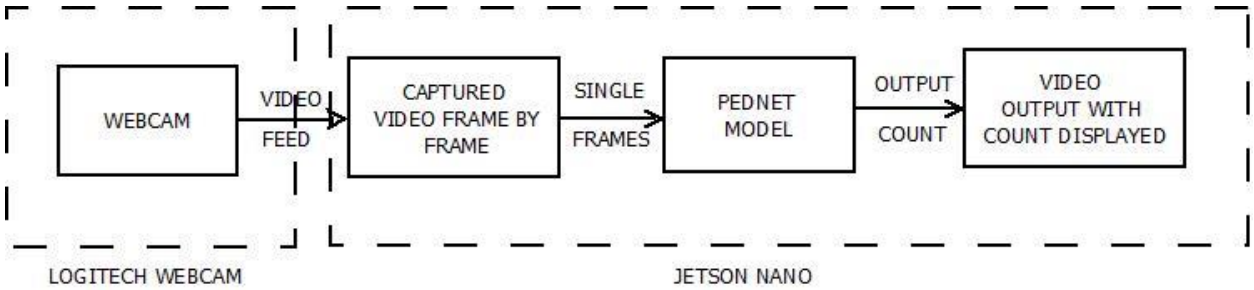

Fig 2.1 Proposed flowchart depicting flow of the implementation and working

3. Results

Regression model was trained by extracting multiple features from the images. The discovery of the immanent importance of various features for crowd estimation was possible by the model.

Initially 3 features- Density, number of Edge-count Pixels and Minkowski Dimension were extracted from the image. The results obtained from the 3 feature model were not very accurate. Another 3 features-determinant, trace and first Eigen value of the Gray Co-occurrence Matrix were added. Hence a total of 6 features were extracted from an image.

From Table 3.1 and Table 3.2 it can be seen that the best chosen models for 3-features and 6-features model are Gaussian Exponential GPR and Gaussian Process Squared Exponential Regression respectively.

Definitions of Performance parameters:

1. RMSE is the Root Mean Square Error, which gives us the absolute measure of the fit
2. R^2 gives us the relative measure of fit
3. Mean Absolute Error (MAE) is a loss function used for regression. The loss is the mean over the absolute differences between true and predicted values.
4. Mean squared error (MSE) measures the mean squared error. MSE tends to punish large errors more because it squares all the errors.
5. Training time is the time taken to train the model in seconds.
6. Prediction speed is the time taken to make predictions on the observations.

Both Training time and Prediction time were measured on a CPU, and are important in their relative values, as these will not be the time durations if they were used on a GPU.

MODEL TYPE	RMSE	R^2	MSE	MAE	PREDICTION SPEED(obs/sec)	TRAINING TIME(sec)
Linear Regression Linear	6.3662	0.23	40.529	5.2527	29000	8.6137
Linear Regression Interaction Linear	6.3729	0.23	40.614	5.2639	49000	0.67019
Linear Regression Robust Linear	6.3655	0.23	40.52	5.2383	72000	1.6627
Stepwise Linear Regression Stepwise Linear	6.3662	0.23	40.529	5.2527	79000	1.5037
Tree Fine Tree	7.4789	-0.07	55.934	6.1067	16000	3.0866
Tree Medium Tree	6.5904	0.17	43.434	5.3775	27000	0.12701
Tree Coarse Tree	6.3885	0.22	40.813	5.1778	180000	0.17999
SVM Linear SVM	6.3899	0.22	40.831	5.2196	32000	2.9629
SVM Quadratic Svm	6.4226	0.21	41.25	5.2491	20000	77.416
SVM Cubic SVM	10.502	-1.1	110.28	8.2646	72000	83.733
SVM Fine Gaussian SVM	6.5315	0.19	42.66	5.2844	100000	0.21503
SVM Medium Gaussian SVM	6.3705	0.23	40.584	5.1519	110000	0.18514
SVM Coarse Gaussian SVM	6.4078	0.22	41.059	5.2532	110000	0.17312
Ensemble Boosted Trees	6.3726	0.23	40.609	5.1277	15000	2.3453
Ensemble Bagged Trees	6.3553	0.23	40.39	5.2147	22000	0.93625
Gaussian Process Squared Exponential Regression	6.3152	0.24	39.882	5.1945	40000	10.208
Gaussian Matern 5/2 GPR	6.3069	0.24	39.776	5.1829	17000	6.1816
Gaussian Exponential GPR	6.2628	0.25	39.223	5.1235	53000	6.2971
Gaussian Rotational Quadratic GPR	6.2939	0.25	39.613	5.1695	41000	12.454
Gaussian PCA	6.2945	0.25	39.621	5.1868	18000	5.253

Table 3.1 Regression model using 3 features

MODEL TYPE	RMSE	R^2	MSE	MAE	PREDICTION SPEED(obs/sec)	TRAINING TIME(sec)
Linear Regression Linear	17.245	-4.55	297.4	14.283	18000	5.3282
Linear Regression Interaction Linear	12.664	-1.99	160.37	10.08	24000	0.86881
Linear Regression Robust Linear	17.224	-4.54	296.67	14.229	42000	1.2154
Stepwise Linear Regression Stepwise Linear	12.689	-2	161.01	10.127	41000	2.7186
Tree Fine Tree	4.1317	0.68	17.071	3.2998	9800	1.5808
Tree Medium Tree	3.6899	0.75	13.615	2.9851	16000	0.16447
Tree Coarse Tree	3.7126	0.74	13.784	3.0479	110000	0.17738
SVM Linear SVM	3.5114	0.77	12.33	2.8011	22000	1.1491
SVM Quadratic Svm	3.4573	0.78	11.954	2.7611	12000	0.63778
SVM Cubic SVM	3.5778	0.76	12.8	2.865	79000	1.8094
SVM Fine Gaussian SVM	4.9602	0.54	24.604	3.7767	75000	0.3052
SVM Medium Gaussian SVM	3.6118	0.76	13.045	2.9071	78000	0.22071
SVM Coarse Gaussian SVM	3.514	0.77	12.348	2.8053	55000	0.17106
Ensemble Boosted Trees	3.8609	0.72	14.906	3.0249	17000	1.2029
Ensemble Bagged Trees	3.6756	0.75	13.51	2.9571	12000	0.6685
Gaussian Process Squared Exponential Regression	3.4115	0.78	11.638	2.7747	26000	13.128
Gaussian Matern 5/2 GPR	3.4221	0.78	11.71	2.7915	11000	9.8316
Gaussian Exponential GPR	3.418	0.78	11.683	2.7888	36000	9.7226
Gaussian Rotational Quadratic GPR	3.4355	0.78	11.802	2.8025	31000	19.731
Gaussian PCA	7.0975	0.06	50.374	5.6845	11000	8.4353

Table 3.2 Regression model using 6 features

The results obtained for the regression models using 3 and 6 features are shown below:

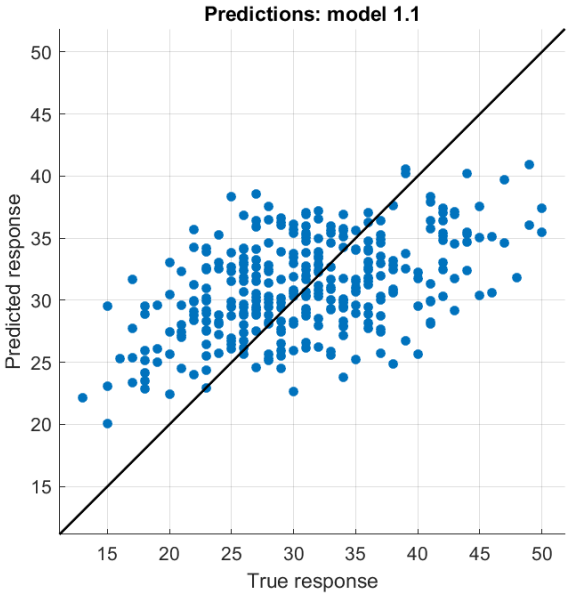

Fig 3.1 Regression model using 3 features

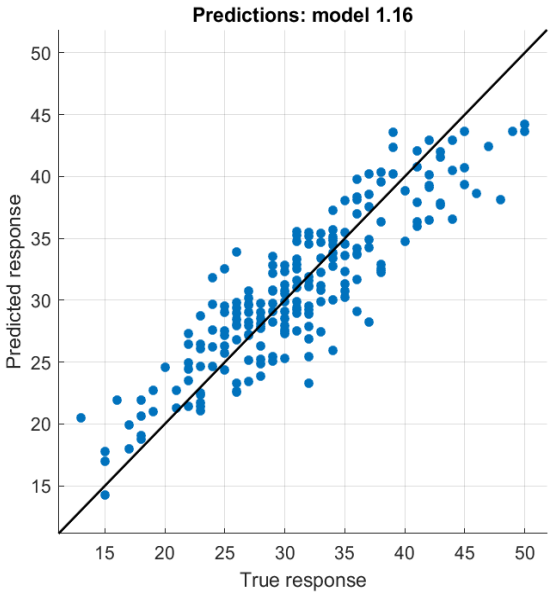

Fig 3.2 Regression model using 6 features (Predictions model)

The regression model using 6 features gave us more accurate results compared to the 3 feature model.

The aim of the regression model was to obtain a best fit line for the count data. The best fit line is the one for which total prediction error (all crowd count data points) are as small as possible. Error is the distance between the point to the regression line. As seen in Fig 3.1 and 3.2, the correlation between the predicted response and true response is better for the 6-feature model. The combination of 6 features covered more properties of the image as compared to the 3-feature model to produce a well-trained regression model.

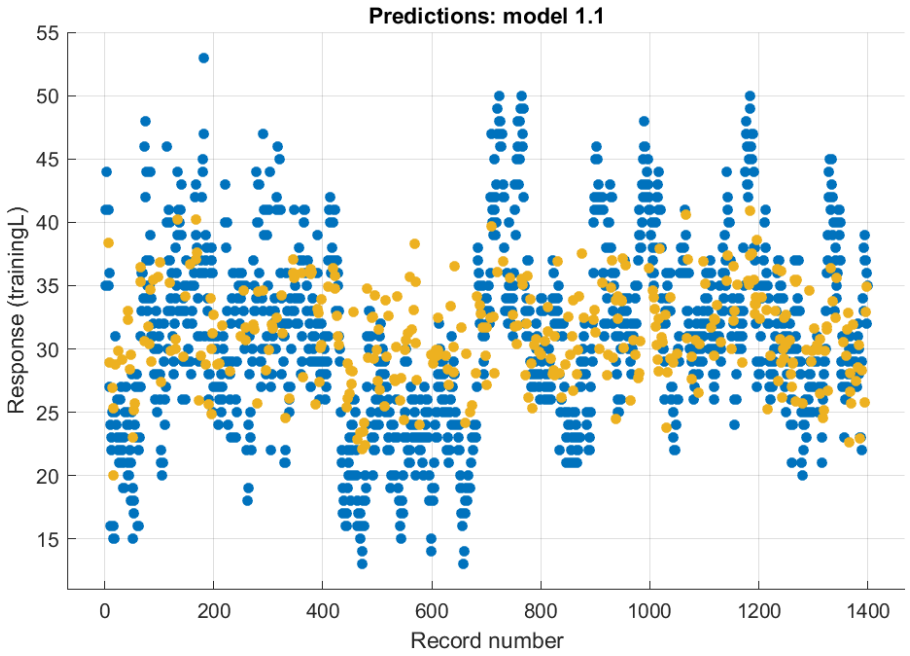

Fig 3.5 Response plot during training for a regression model using 3 features

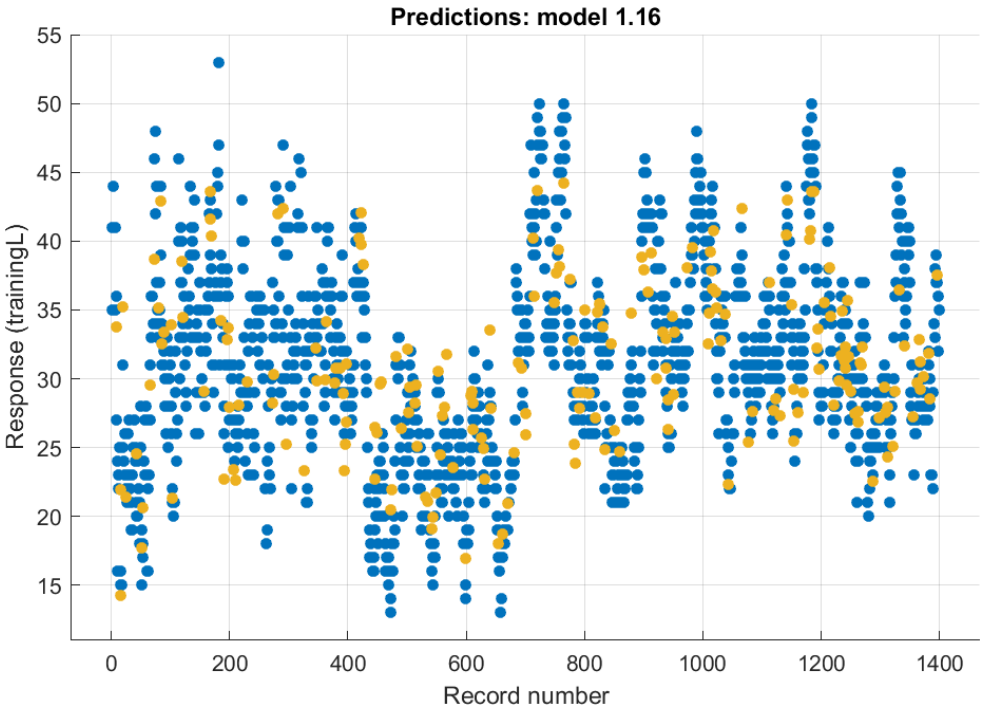

Fig 3.6 Response plot during training for a regression model using 6 features

Response plot is a plot of the response vs Record number. A better fit between the target and the output was achieved for the 6 feature model, as more number of points overlap.

With higher number of features extracted from the image, the accuracy of the crowd count obtained was better. The best chosen models for 3 features model and 6 features model were performed on 600 test images using MATLAB and an accuracy of 90.1807% was obtained for the 6 features model.

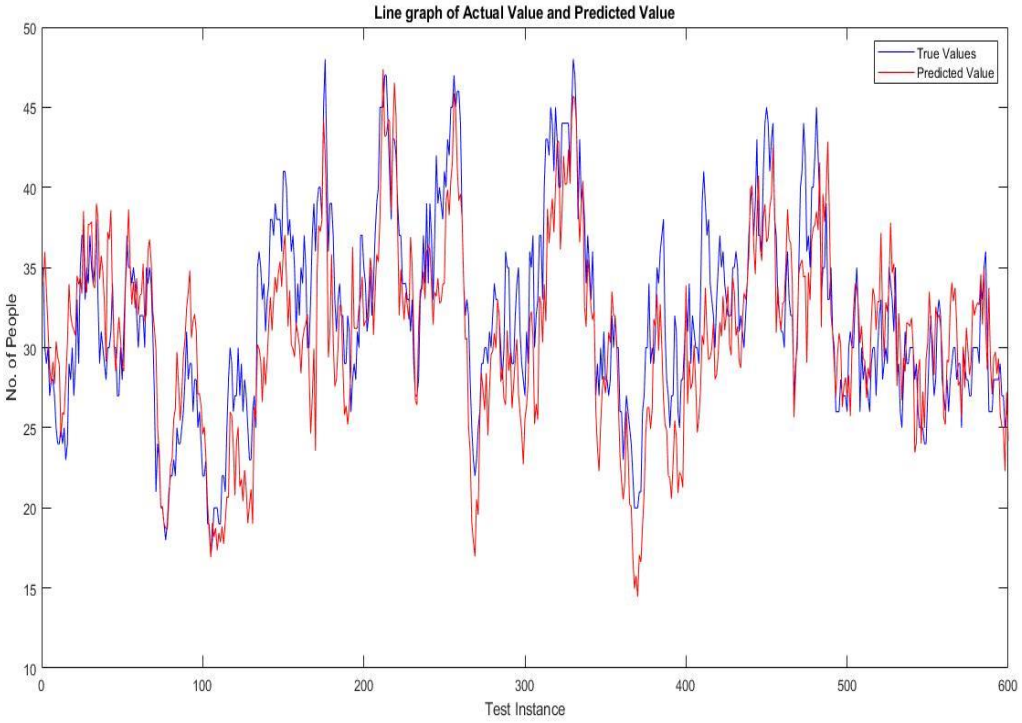

Fig 3.7 Line graph of actual versus predicted values for the 6 feature regression model

Fig 3.7 depicts the line graph of the actual value versus predicted value for the 6-feature model. The values predicted are shown in red, and almost entirely overlap the True values which are in blue. However, this model did not work well with real world images as it was overfit. Overfitting is a modeling error that occurs when a function is too closely fit to a limited set of data points.

Hence, another model using CNN was implemented and the following results were obtained as shown below:

Image depicting people in the mall (procured from the Mall Dataset) (Faster RCNN output)

Image depicting the output as obtained via PedNet

Conclusions and future scope

The performance of the CNNs on the dataset in real world scenarios needs to be evaluated. However, preliminary results show that CNN will perform better than the regression models worked on and hence it is the right choice for our project. According to the results obtained, it can be concluded that regression is also very time consuming ie. to extract more than 6 features for each frame in the video feed would be an arduous task.

The model has yet to be deployed on the Jetson Nano ie. live video feed has to be taken from the Logitech webcam, the detection and counting algorithm must be run and the number of people present must be found out.

References

1. Crowd Density Estimation Using Multiple Features Categories and Multiple Regression Models, Ahmed F. Gad.
2. SCAR: Spatial-/channel-wise Attention Regression Networks for Crowd Counting, Junyu Gao.
3. PedNet: A Spatio-Temporal Deep Convolutional Neural Network for Pedestrian Segmentation, Mohib Ullah.
4. Deep Metric Learning for Crowdedness Regression, Qi Wang, Senior Member, IEEE.
5. Feature Mining for Localised Crowd Counting, Ke Chen.
6. Improved Crowd Counting Method Based on Scale-Adaptive Convolutional Neural Network, Jun Sang.
7. Residual Regression with Semantic Prior for Crowd Counting, Jia Wan.
8. People Counting in Dense Crowd Images Using Sparse Head Detections, Mamoona Birkhez Shami.