CAB420 Assignment 2

Crowd Counting

Team Members:

Name	**Student Id.**
Duane Johnston	n10584269
Mitchell Barron	n10214640
Nihar Rupareliya	n10355243

Due: 6th June 2021

Contents

Introduction

The importance of crowd counting comes from the need to improve safety in any area or event that may quickly become overcrowded. This has become more important since COVID-19 has shaped the way we need social distancing to protect everyone to stop the spread of the virus. Crowd counting aims to count the number of people in an image. The primary motivation for this report is how it can be used to improve public bus networks in Brisbane. Unlike the train network, where passengers would tap on at the station, the bus network allows passengers to tap on once they are already on the bus. However, it does not count potential passengers waiting at the bus station. This could be an issue, especially for the (near-certain) hosting of the 2032 Olympic games in Brisbane and the uncertainty in the movement of such many national and international visitors. The possibility of using cameras to count potential passengers at bus stations could allow for an increased frequency in busses to keep up with demand. While no data sets on Brisbane bus stations are available, to ensure the work is comparable to systems here, a collection of images from a bus station in Beijing will be used.

Related Work

Crowd counting attracts many different techniques that fall into three categories: counting by detection, clustering, and by regression (Chen, Loy, Gong, & Xiang, 2012). For our methods, we will be using different types of counting by regression as the model aims to learn a direct mapping between low-level features and a number of people in the image (Chen et al., 2012). This approach is better suited for our project because it is not computationally expensive and works well in large crowds with low frame rate video.

Two crowd counting articles were reviewed using the Beijing-BRT-dataset we decided to use for our crowd counting task:

A Deeply Recursive Convolutional Network for Crowd Counting

The article written by Ding et al. proposed a Deeply Recursive CNN for crowd counting, which used the Beijing-BRT-dataset alongside others. They also introduced a new dataset for crowd counting, which had images from a public scene containing 1,280 images with 16,795 pedestrians labelled (Ding, Lin, He, Wang, & Huang, 2018). The model they developed was based on the ResNet model because of its high performance and fast convergence with image classification and regression tasks (Ding et al., 2018). However, they decided to build a deeply recursive model to reduce significantly the number of parameters required to train the network and improve performance. The model architecture can be seen in Fig. 1.

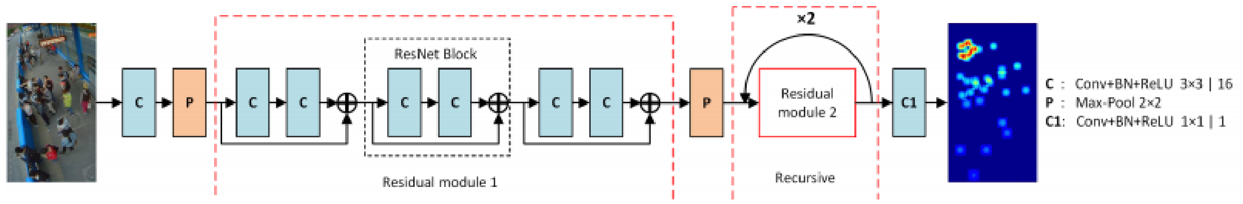


Figure 1 Deeply Recursive ResNet (DR-ResNet) architecture (Taken from Ding et al.)

The model takes in an image and performs a 3x3 Convolution with 16 filters, Batch Normalisation and ReLU activation, followed by a Max Pooling of size 2x2. It is then passed through' Residual Module 1', which contains 3 ResNet Blocks in series. Each ResNet block has 2 Convolutional layers with a skip connection. The skip connection allows for data to backpropagate easily through a large network. After Residual module 1, there is another Max Pooling, same as earlier before entering Residual module 2. Residual module 2 is where the model gets its name because it runs the module recursively. Residual module 1 would be represented by y = f(x), Residual module 2 is y = f(f(f(x))). Finally, the output of Residual module 2 is a 1x1 Convolution using Batch Normalization and ReLU. The output of this can then be regressed to count the number of people in an image.

The table below (Fig. 2), taken from the article, shows that they were successful in their hypothesis that the Recursive module implemented had significantly lowered the error, compared to models with the same or slightly higher. The only one which beat the recursive model in terms of error was the ResNet-26 (ResNet with 26 convolutional layers) which contained double the number of parameters used by the recursive model.

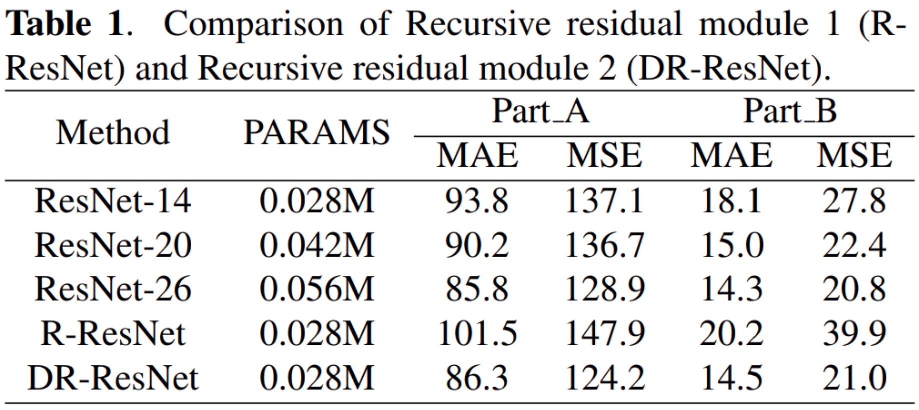


Table 1. Comparison of Recursive residual module 1 (R-ResNet) and Recursive residual module 2 (DR-ResNet).

Method	PARAMS	Part_A		Part_B	
		MAE	MSE	MAE	MSE
ResNet-14	0.028M	93.8	137.1	18.1	27.8
ResNet-20	0.042M	90.2	136.7	15.0	22.4
ResNet-26	0.056M	85.8	128.9	14.3	20.8
R-ResNet	0.028M	101.5	147.9	20.2	39.9
DR-ResNet	0.028M	86.3	124.2	14.5	21.0

Figure 2 DR-ResNet Parameter and Performance results (Taken from Ding et al.)

Learning a deep network with cross-hierarchy aggregation for crowd counting

This article is written by Guo et al. and proposes using cross-hierarchy aggregation for crowd counting applications (Guo, Zeng, Hu, Phoummixay, & Ye, 2021). It also uses the Beijing BRT dataset alongside other typical crowd counting datasets. Furthermore, this paper proposed building a model that combines the local and global hierarchical features of an image to increase the model's performance. The model architecture can be seen in Fig. 3 and explains how they implement the proposed CHA module.

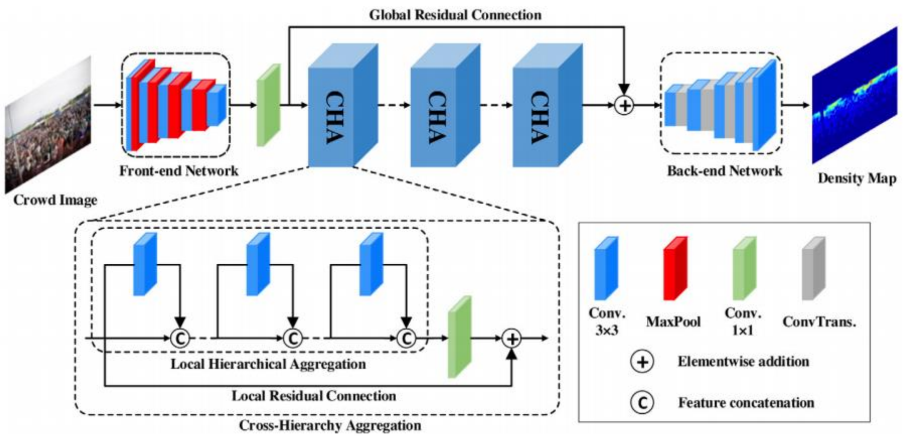


Figure 3 CHANet Model architecture (taken from REF)

The design uses the first thirteen layers of the VGG-16 model as the front end due to the strong representation ability and strong transfer capability (Guo et al., 2021). The design uses a 1x1 Convolutional layer between the Front-end Network and the Cross-Hierarchy Aggregation (CHA) modules. These CHA modules have been broken down in Fig. 3 to explain better how they work. The CHA modules use a Local Residual Connection bypass to channel the local features through the network and find new local features through the 3x3 Convolutional blocks and skip connections. The outer Global Residual Connection acts similarly to the Local Residual Connection by adding the result of the front-end network to the end of the three series-connected CHA modules to ensure the global hierarchical features are not lost. This is then followed by a back-end decoder sequenced with convolutions and transposed convolutions to generate a corresponding density map (Guo et al., 2021).

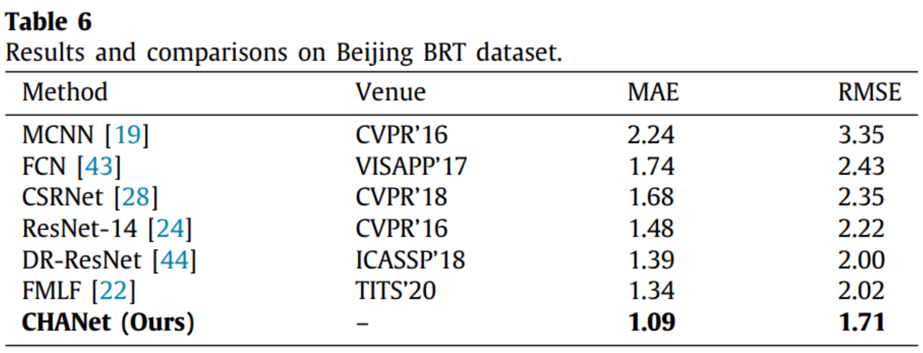


Table 6
Results and comparisons on Beijing BRT dataset.

Method	Venue	MAE	RMSE
MCNN [19]	CVPR'16	2.24	3.35
FCN [43]	VISAPP'17	1.74	2.43
CSRNet [28]	CVPR'18	1.68	2.35
ResNet-14 [24]	CVPR'16	1.48	2.22
DR-ResNet [44]	ICASSP'18	1.39	2.00
FMLF [22]	TITS'20	1.34	2.02
CHANet (Ours)	–	**1.09**	**1.71**

Figure 4 CHANet Results (Taken from Guo et al.)

As seen by the table above in Fig. 4, The results and comparisons are taken from the Beijing BRT dataset to compare to our models. The CHANet came through on its hypothesis that it would beat comparable state-of-the-art models available. They use both Mean Squared Error (MSE) and Root Mean Square Error (RMSE) as error functions, both of which come in lower than their competing models.

Data

The dataset used for our crowd counting model was the Beijing BRT dataset (XMU-smartdsp, n.d.). This dataset includes 1,280 images with 16,795 labelled pedestrians for crowd analysis. The dataset is split into 720 images for training and 560 images for testing, so about a 60:40 train-test split. The dataset also comes with a set of .mat files with the same naming convention as the image they refer to. In addition, the file contains a matrix where each row contains the i-th persons [x, y] location in the image so that a density map can be made.

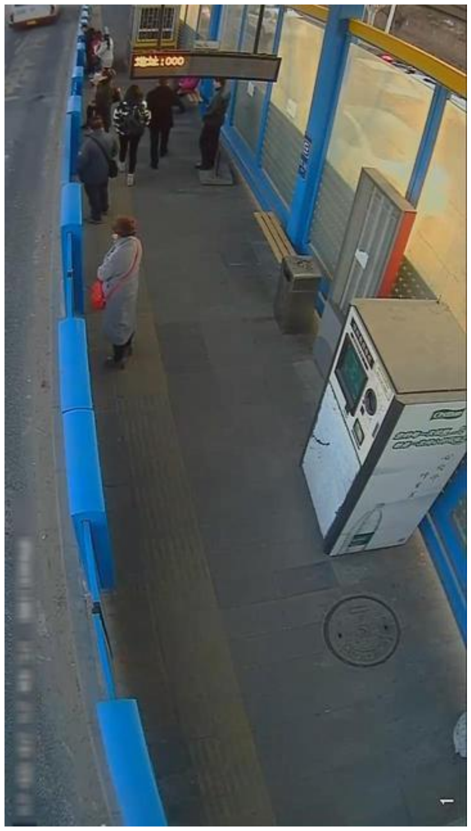


Figure 6 Image from Beijing BRT dataset

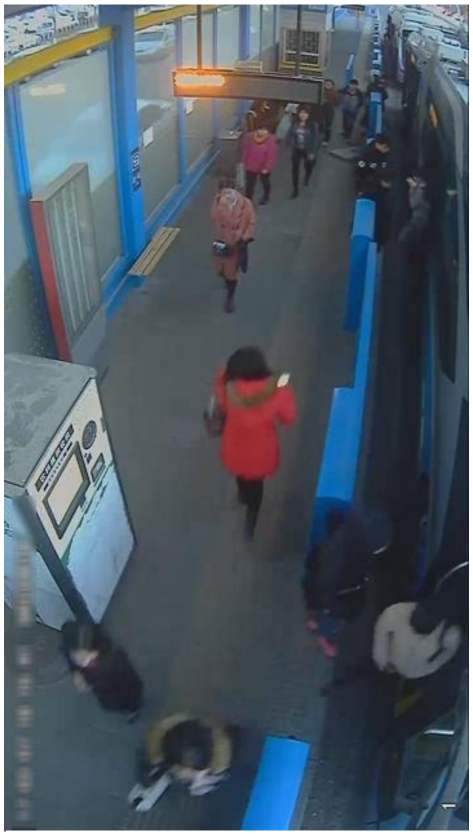


Figure 5 Image from Beijing BRT dataset

Methodology

Model 1: Auto-encoder into a Linear Regression Model

The first model created was an autoencoder DCNN whose embedding was used to create a linear regression model. This DCNN was trained as a multitask, with one output as the original image and the other as the count values. These two outputs were chosen to ensure the embeddings created by the encoder contained information on the initial input that was filtered to show the most relevant information to describe the number of people. The training set was split 6:4 into training and validation for this model, and the model allowed for horizontal flips for the inputs. Given cameras facing either way at the station, this augmentation will not significantly change the image.

The DCNN model consisted of three convolution blocks. These blocks consisted of two 2D convolution layers, a ReLU activation layer and a Max Pooling layer. The number of filters was 64,32, and 16, and the max-pooling was 4x3, 4x3 and 2x2. The reason for the size of the max-pooling was the belief that vertical rectangles would better pick up the pattern we were identifying (people standing). The occasional person is sitting in the images, but sitting upright still makes the individual appear more like a rectangle. Also, the images are enormous, so the large size of the pooling helps reduce the size of the embeddings. This is where the model diverges.

For the count data, the output is flattened. The embedding size after this point is a 6400 vector. Given the data set only consists of 1,280 images, having such an extensive embedding will cause the linear regression to fail as fewer samples than predictors. Therefore, it was decided to put this embedding through a dense layer to a 32-length vector. This may still be on the larger size for the given data set but should allow the linear regression to fit while retaining helpful information on the number of people in the image. This 32-vector was put into another dense layer of length 1 to produce the count output.

The up sampling is done in three blocks, each with two 2D Convolution layers and an up-sampling layer. The number of filters for this was 32,32, and 64. In addition, the up sampling was done in the inverse of the max-pooling layers. This ensured the image got to the same length and width. Finally, there was an additional 2D convolution layer with three filters, so the image was the exact dimensions as the input images.

The 32-length embeddings were then taken out and put into a linear regression model. This was then tested to search ridge regression and Lasso ridge regression to find the best regression model possible.

Model 2 DCNN Model

The second model is a forward fed DCNN model utilizing attention. The same data split, and augmentations were used in model 1. The model consists of four convolution blocks with two attention blocks, one after the second convolution block and one after the third. The convolution blocks were the same as in model one, consisting of two 2D convolution layers, a ReLU activation layer and a Max Pooling layer. The number of filters used was 16,32,64 and 128, with the max-pooling layers being 4x3, 4x3, 2x2 and 2x2.

The attention blocks were basic, consisting of a 1x1 2D convolution with a ReLU activation into a sigmoid activation layer. This sigmoid activation was then used multiplied by the input of the activation layer, and this was the output of the activation.

After going through the six blocks, the output is flattened and put into two dense layers, a 64-dense embedding into one dense embedding to get the final count.

It was found that including a third attention block did not help increase the accuracy at all. This is believed to be because the number of filters after the first convolutional block is minimal and has not differentiated a lot between the images. To allow the attention blocks to focus on essential parts, it was decided to leave out the coalition between the first and second convolution blocks. This allowed the model to split the images through two convolution layers before the most critical areas were focused.

Model 3 Congested Scene Recognition (CSRNet)

Data Pre-processing:

Using the following Beijing-BRT-dataset, we convert the provided ground truth into the density maps for the pre-processing. First, we build a kdtree of the head annotations for the given image inside the dataset. After that, we get the location of that sparse matrix and generate a 2D density map by passing it through the Gaussian Filter. Finally, the sum of all the cells inside the density map gives us the actual count of the crowd in the image.

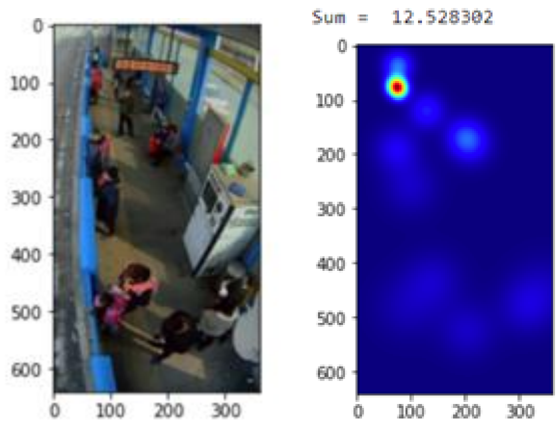


Figure 7: Actual image and the density map with count

Model training and processing:

Our model utilizes the Convolutional Neural Network to get the image that we put to its respective density map. Thus, the model learns from many different-different data, and there is no loss regarding the image resolution (no resize/reshape is required while we are inferencing). The module architecture considers the input image of (x, y,3) and the density map (x/8, y/8,1).

The front-end has 13 trained layers of the VGG16 model (10 convolution layers and 3 Max Pooling layers). The back-end part has Dilated Convolution layers. The dilation rate maximum accuracy was found to be two, as suggested by Li et al.. As VGG16 does not have any Batch Normalization layers, we design our custom VGG16 model and import the trained weights of VGG16 to this model.

Keras library does not allow the variable size of the input to be trained in the same batch. We could combine all the image having the same image shape/ dimension and train them as a batch, but the dataset does not contain many images having the same size, and thus, it will be hard to make such a batch. We, therefore, built a custom data generator in Keras to train.

The dataset is trained with the architecture for 200 epoch, and other hyperparameters were kept identical to the model by Li et al..

After training, we save the weights and the trained model for further analysis (see the evaluation section below).

Evaluation and Discussions

Model 1

The first model that was fit to the embeddings was a standard Linear Regression model for understanding how well the regression had a fit. The regression has fit a bit too well, with an R-squared of 0.975 and an adjusted R-squared of 0.964. Of the 32 'predictors' from the embedding, 9 of them have been deemed useless and have coefficients of 0. Of the remaining 23 predictors with coefficients, eight were significant. The qq-plot has a slight problem, with a small curve noticeable throughout the center residuals [Figure B]. They all stay close to the standard line, which the only significant exceptions being the four greatest

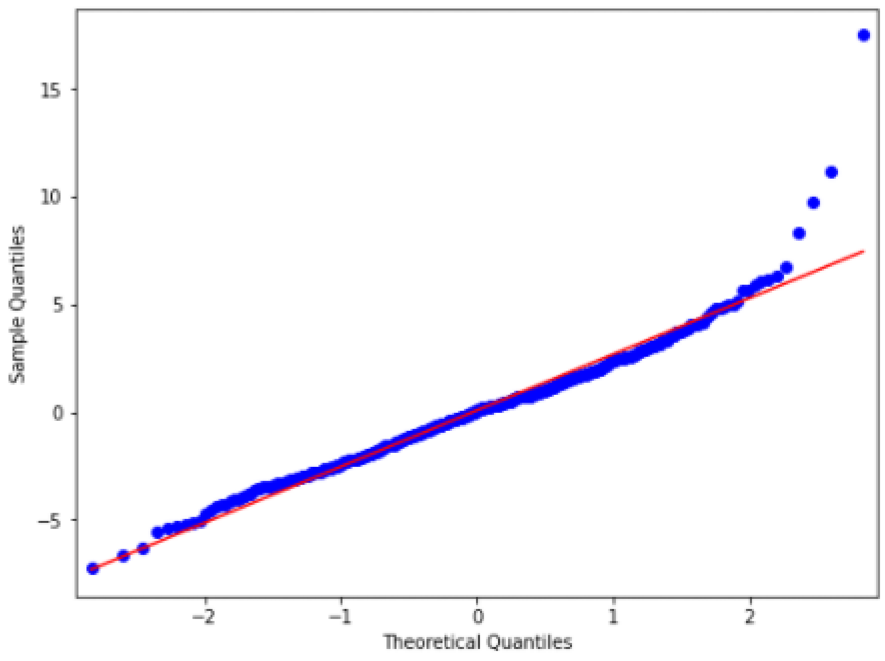


Figure 8 Predicted versus Actual Plots for 3 sets using the Attention Model

residuals, which deviate significantly away from the rest of the points. Apart from those four points, the model fits the data well. While the number of predictors is small enough to go through and remove them by hand, a Lasso and Ridge regression will be used instead to save time.

There came problems when attempting to use Lasso and ridge regression with the embeddings. For Lasso regression, it was found that the model was unable to fit the data correctly. It is unclear exactly what has gone wrong. Then when the values of alpha are above .0001, the regression begins overfitting to the training set. At values less than this, the Lasso regression had problems in converging correctly. This was seen in the curves lines for the alphas being very jagged and not smooth like expected. Due to these errors, it was decided to ignore Lass regression for this model.

For the ridge regression, it was found that the value of alpha needed to minimize the validation RMSE to two decimal places was 511.64. This was found through a grid search of all integers from 1 to 1000, then a more refined search around 500 where the curve's minimum was. This model was chosen for the final model. This model had a training RSME of approximately 2.707, a validation RSME of 5.258 and a testing RSME of 4.583 [Figure C]. Surprisingly, the testing set did better than the validation. When comparing the predicted versus actual plots of the two, the testing set had fewer large values (above 50) than the validation set. Given these are going to having higher chances of significant errors, this makes sense. There appears to be a slight problem with overfitting due to the lower error in the training set, but it does not seem overly extreme.

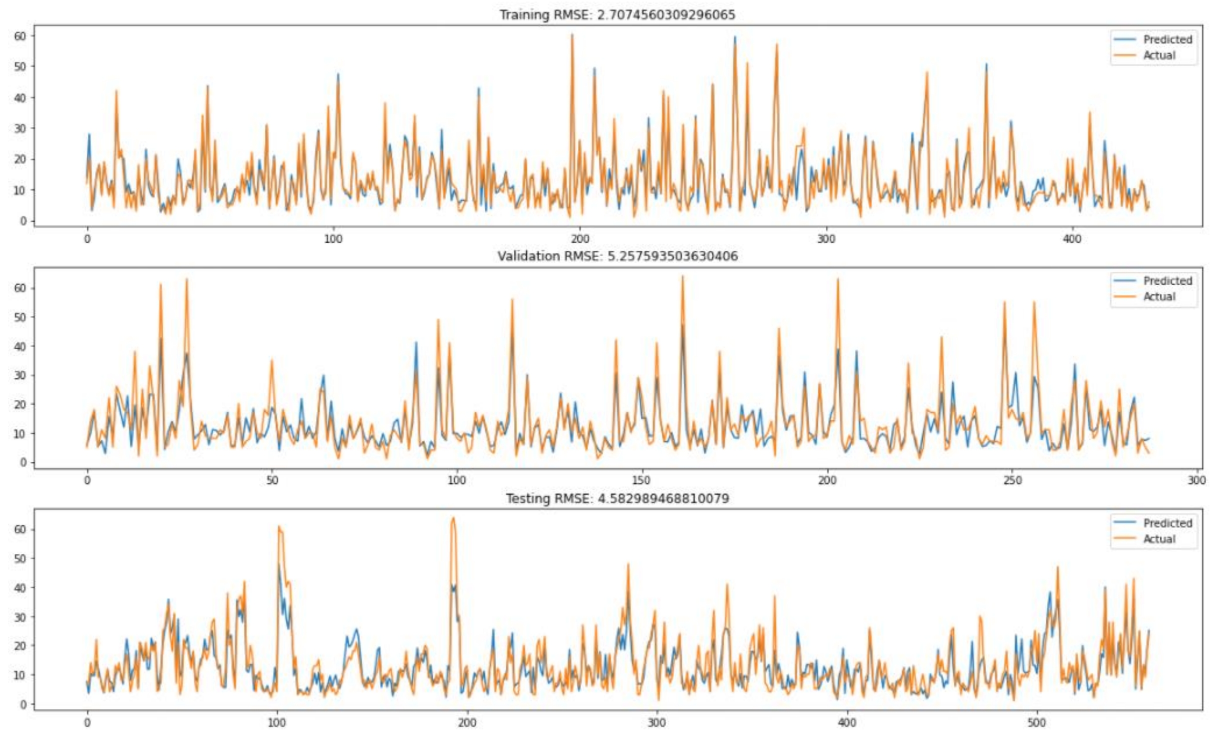


Figure 9 Predicted versus Actual Plots for 3 sets using the Ridge Regression

Model 2

The model had a training RMSE of 2.648, a validation RMSE of 5.471 and a testing RMSE of 4.428 [Figure A]. Again, the testing set performed better than the validation due to many high-count images in the validation set. This model again shows signs of overfitting the data, but it does not appear significantly large.

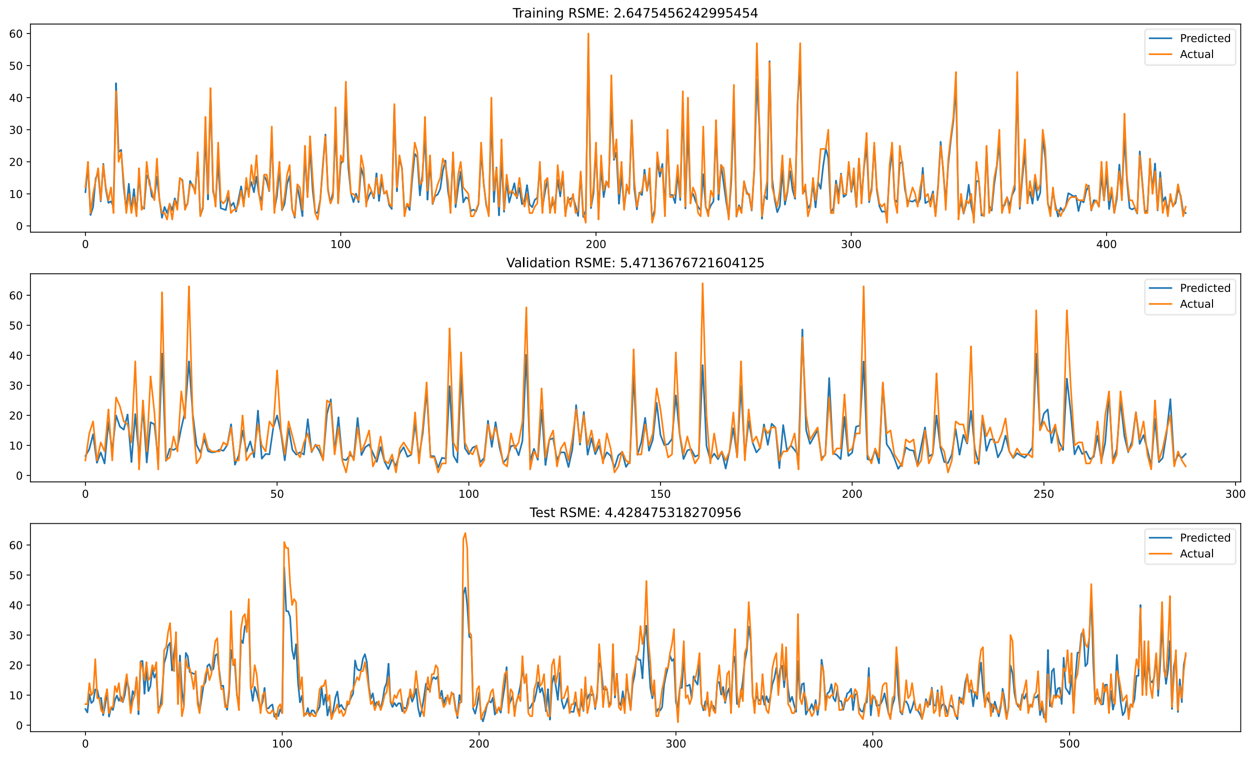
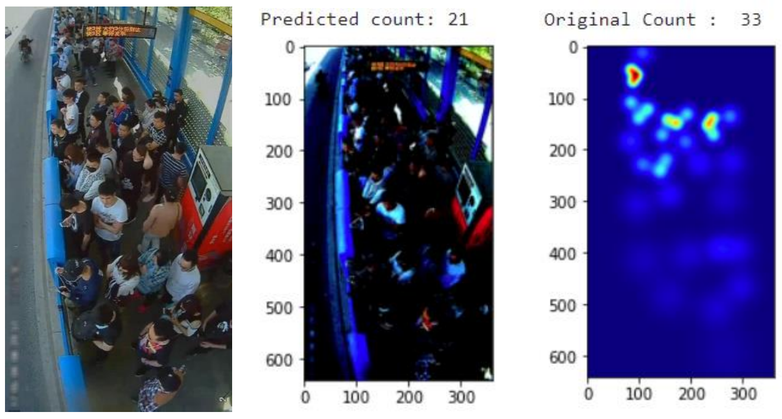


Figure 10 Predicted versus Actual Plots for 3 sets using the Attention Model

Model 3:



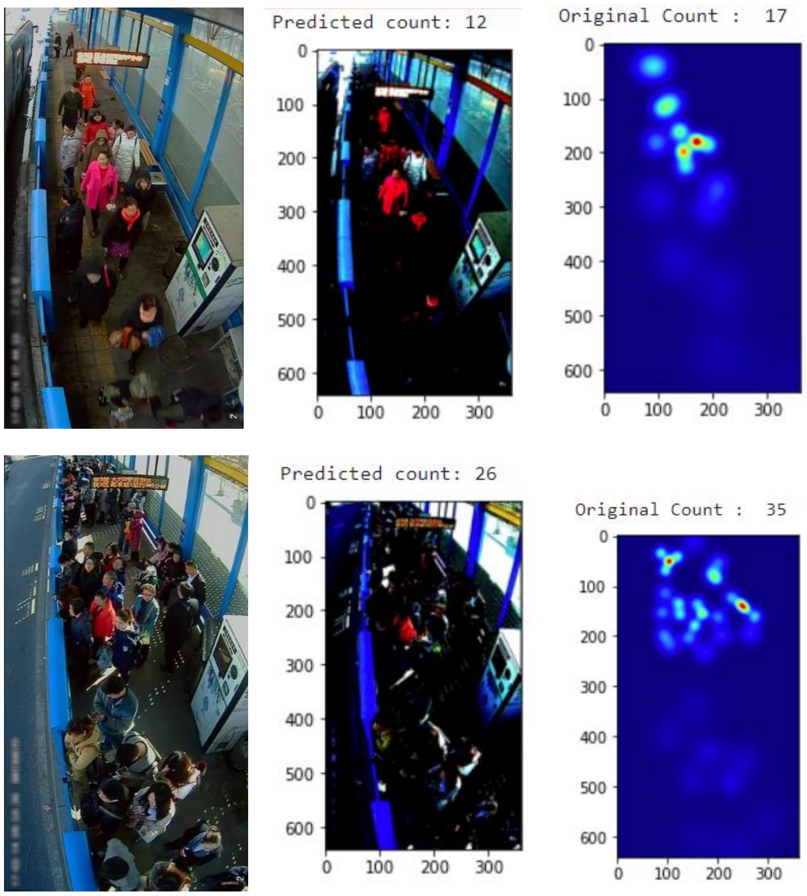


Figure 11 Original image| Predicted Crowd | Original Count

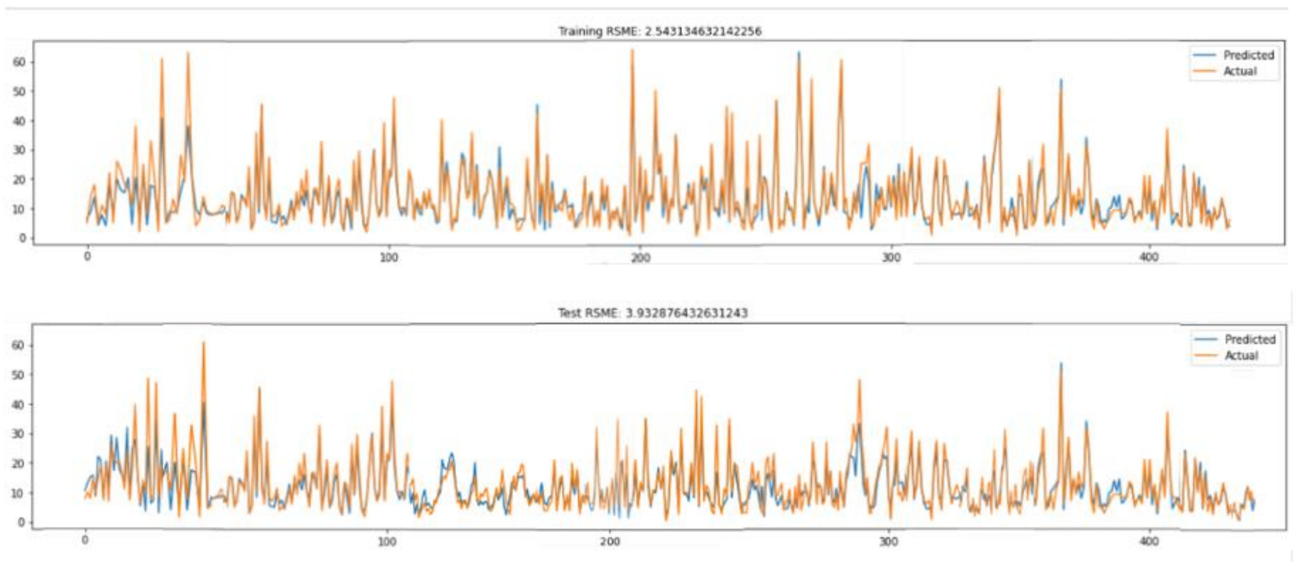


Figure 12 RMSE for train and test set

The model was tested on three different conditions as seen in the image having High Crowd density, low crowd density and high crowd with sunlight. It had an RMSE of 2.5431346321422562 on the training set with 400 images, and the RMSE of the test set was 3.9328764326312431. The mean absolute was also calculated our prediction, which was found to be 58.76437532642134. It was found that the density chart prediction was slightly more accurate when the crowd density was less than when more people were present. Though not perfect, the density graph seems to be doing an excellent job despite the sunlight and other factors.

CSRnet publication Citation: (Li, Zhang, & Chen, 2018)

CSRNet GitHub citation: (Neerajj, 2019)

Model Comparison

Model	**Train RMSE**	**Val RMSE**	**Test RMSE**
Auto-encoder into a Linear Regression Model	2.707	5.258	4.583
DCNN model utilizing attention	2.648	5.471	4.428
Congested Scene Recognition (CSRNet)	2.543	N/A	3.932

Table 1: RMSE comparison of the three models

All three models trained to similar scales, with only Model 3 having a significant difference in training and test RSME [Table 1]. The model 3 result cannot be accurately compared to models 1 and 2 due to the different sets used. Model 1 and 2 used the same training, validation, and testing sets, but three had to be used on a portion of the original sets due to some limitations model. When directly comparing models one and two, they both trained about and each other, with difference RSME no greater than 0.2 between any of the sets. Model 2 has a better test set RSME, so that this one would be chosen out of those two. Comparing this to model 3 significantly has a better test RMSE; however, they cannot be compared directly as they use different data sets.

The biggest problem between all three models was an underestimation of large counts. In models one and two, the predictions rarely got above 40. Although model 3 does do a better job at this, it has still fallen well short on occasion. The believed problem for this is the size difference between two people because of the camera's location. An example of this can be seen in figure X, with a large group at the far end of the station. Visually, these people take up far less room on the screen than those standing under the camera. It is believed that this difference in size tricks the system as there are no built-in sections that focus on this. For the objective of identifying images with higher counts, the first two models perform the task poorly. Therefore, it would be recommended to use model 3 for this task.

Conclusions and Future Works

The project's objective was to investigate different methods to count the number of people in an image. The primary goal of this was to use it to identify areas that had many people in a bus network. To do this, three different models were trained and evaluated against each other. Of the three models, the prediction error was comparable. However, two of the models had most of their error occurring in points with large counts, while the other had a more even spread of error across all points. Due to this, the third model, a CSRNet, was recommended.

As noted earlier, a large part of the error is believed to come from the furthest people away from the camera. It would be desirable to try and fix this in future works, as it should greatly help all the models. In addition, trying to work on improving the current model and further extending the analysis of the data is also subject for the future work.

References

Chen, K., Loy, C. C., Gong, S., & Xiang, T. (2012). Feature mining for localized crowd counting. *Bmvc*, *1*, 3.

Ding, X., Lin, Z., He, F., Wang, Y., & Huang, Y. (2018, April). A Deeply-Recursive Convolutional Network For Crowd Counting. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1942–1946.

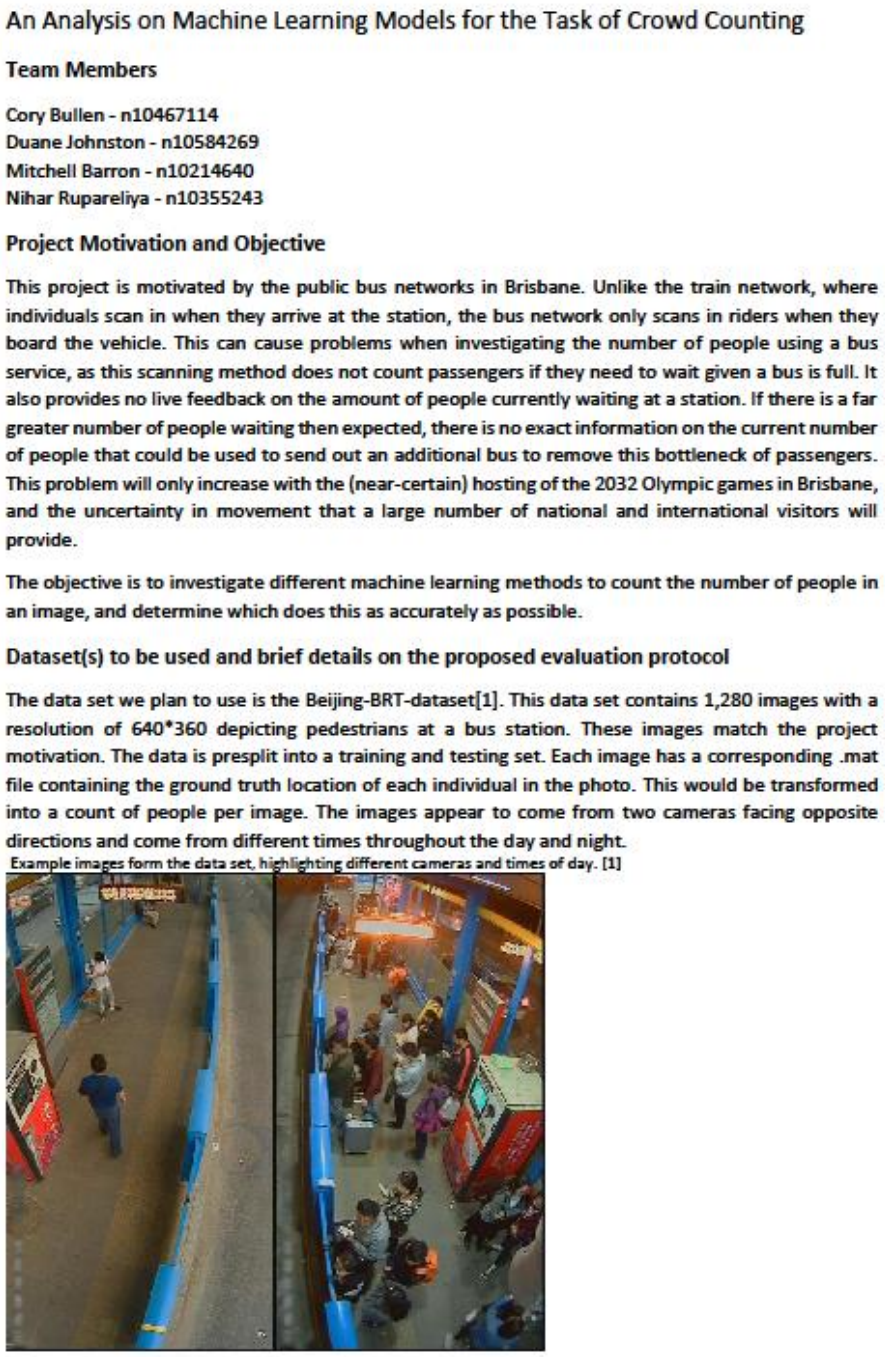
Guo, Q., Zeng, X., Hu, S., Phoummixay, S., & Ye, Y. (2021). Learning a deep network with cross-hierarchy aggregation for crowd counting. *Knowledge-Based Systems*, *213*, 106691.

XMU-smartdsp. (n.d.). Beijing-BRT-dataset. Retrieved 6 May 2021, from GitHub website: https://github.com/XMU-smartdsp/Beijing-BRT-dataset

Appendix A

Name	ID	Contribution
Duane Johnston	n10584269	Introduction, Related Work, Research/ References and Data
Mitchell Barron	n10214640	Model 1 and 2 -Methodology and results, Model comparison, conclusion and future work
Nihar Rupareliya	n10355243	The third model - Methodology and Results/ Conclusion and References

Appendix B



An Analysis on Machine Learning Models for the Task of Crowd Counting

Team Members

Cory Bullen - n10467114
Duane Johnston - n10584269
Mitchell Barron - n10214640
Nihar Rupareliya - n10355243

Project Motivation and Objective

This project is motivated by the public bus networks in Brisbane. Unlike the train network, where individuals scan in when they arrive at the station, the bus network only scans in riders when they board the vehicle. This can cause problems when investigating the number of people using a bus service, as this scanning method does not count passengers if they need to wait given a bus is full. It also provides no live feedback on the amount of people currently waiting at a station. If there is a far greater number of people waiting then expected, there is no exact information on the current number of people that could be used to send out an additional bus to remove this bottleneck of passengers. This problem will only increase with the (near-certain) hosting of the 2032 Olympic games in Brisbane, and the uncertainty in movement that a large number of national and international visitors will provide.

The objective is to investigate different machine learning methods to count the number of people in an image, and determine which does this as accurately as possible.

Dataset(s) to be used and brief details on the proposed evaluation protocol

The data set we plan to use is the Beijing-BRT-dataset[1]. This data set contains 1,280 images with a resolution of 640*360 depicting pedestrians at a bus station. These images match the project motivation. The data is presplit into a training and testing set. Each image has a corresponding .mat file containing the ground truth location of each individual in the photo. This would be transformed into a count of people per image. The images appear to come from two cameras facing opposite directions and come from different times throughout the day and night.

Example images form the data set, highlighting different cameras and times of day. [1]