CAB420 Assignment 2

Group 19

Jacob Woods	n10234853
Flynn Kassulke	n10809511
Jean Terblans	n11246332

1 Introduction

Crowd counting, the task of estimating the number of people in a scene, is a crucial problem with significant applications in areas such as public safety and event management. This project aims to assess and compare performance of three distinct machine learning algorithms in accurately counting the number of people in various scenes. By evaluating these algorithms, we seek to determine their respective strengths and weaknesses in addressing the complexities of crowd counting.

Our research question centres on evaluating which machine learning methods are most effective for crowd counting and under what conditions each algorithm performs best. This involves not only assessing their respective accuracy but also their computational efficiency, robustness to varying crowd densities, and ability to generalise across dynamic environments.

This project builds on previous work in the field of computer vision and machine learning, where many approaches have been proposed for crowd counting. Prior studies have highlighted the challenges inherent in crowd counting, such as occlusion, perspective distortion, and diverse crowd formations. By comparing distinct algorithms, this project contributes to the ongoing research by providing insights into the practical applicability and limitations of these methods.

2 Related Work

Paper [1] explores a real time convolutional network for real time crowd counting. As the model is focused on real time computation/prediction, its goal is to execute quickly. Hence the model is compact and lightweight. What is unique about the model is the use of 3 parallel convolutional layers: "The red layer is designed to pay more attention to large receptive fields … the green one is on behalf of dense crowds and the blue one stands for highly congested crowded scenes". These layers can be seen in paper [1]'s proposed model below in figure 1. Through the parallel computing the model was "fast and accurate" [1]. The resulting model from [1] is 0.07 million parameters and was able to calculate 104.6 frames per second. The model was also able to average approximately the same error calculations (MSE, MAE, etc) as larger parameter models (5.1M+). Overall, this model is highly effective in crowd counting, with extraordinary compute time thanks to its lightweight but still maintains a focus on accuracy.

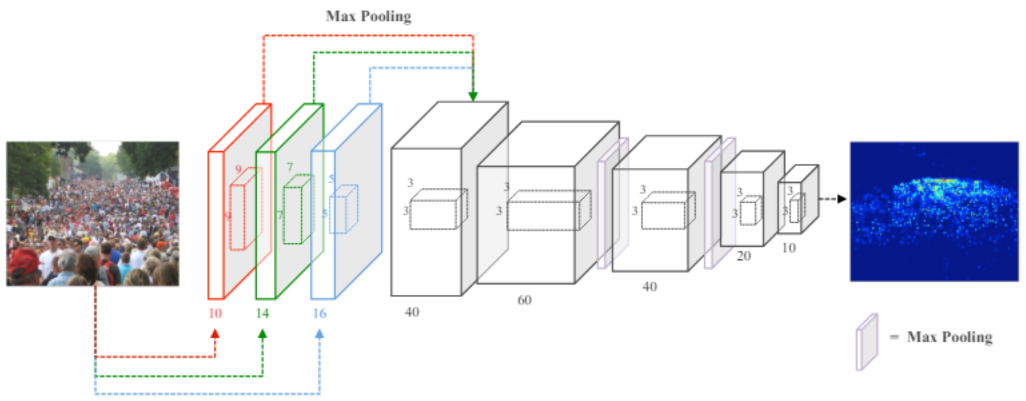


Figure 1: The model architecture proposed in paper [1]. The model uses parallel computing as seen in the red, blue and green lines to adjust for various crowd scenes. Red for large fields, green for dense crowds and blue for highly congested crowds.

Paper [2] explores a deep convolutional network (DCNN) for dense crowd counting called "CrowdNet". The model is focused on accurately estimating the number of people in each image for dense crowds (500+). The proposed model architecture in [2] is similar to [1] in that it has parallel component, however this model's branches are a deep network and a shallow network as seen in figure 2 below. The resultant model size is not discussed with in paper [2]. The resulting model is shown however to have a lower mean squared error than other models focused on dense crowd counting. Although the margin of improvement is rather low and without knowing the size of the model its unknown, the effective performance of the model. Overall, this model is focused on accuracy within dense crowd counting and is able to slightly improve accuracy over other existing models.

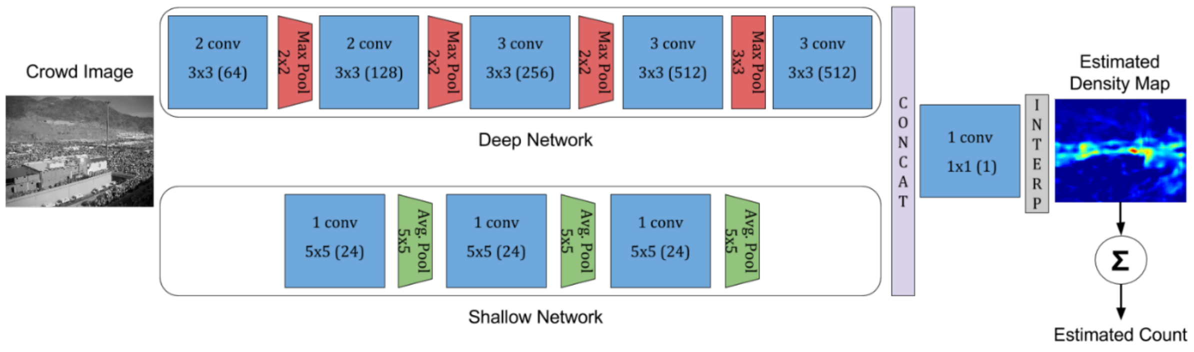


Figure 2: The proposed model architecture in paper [2]. The model has a parallel network, one deep and one shallow that concatenate for the final output.

Paper [3] explores "Cross-scene Crowd Counting via Deep Convolutional Neural Networks". The model is purposed towards handling unseen target scenes without requiring new annotations. The model training method can be seen in Figure 3 below: first a CNN is trained before it is fine-tuned in the new target scenario. The resultant model is highly accurate in its new target scene often producing lower mean squared error values than other models. This is due to its dual learning objectives (crowd density and crowd count) and the fine-tuning process that adapts the pre-trained CNN to the specific characteristics of the target scene. However, due to the model's training/fine tuning method it will be rather large in size although paper [3] does not specify. Overall, the proposed model in [3] is ideal for unseen target scenarios where parameters are considered unimportant compared to accuracy.

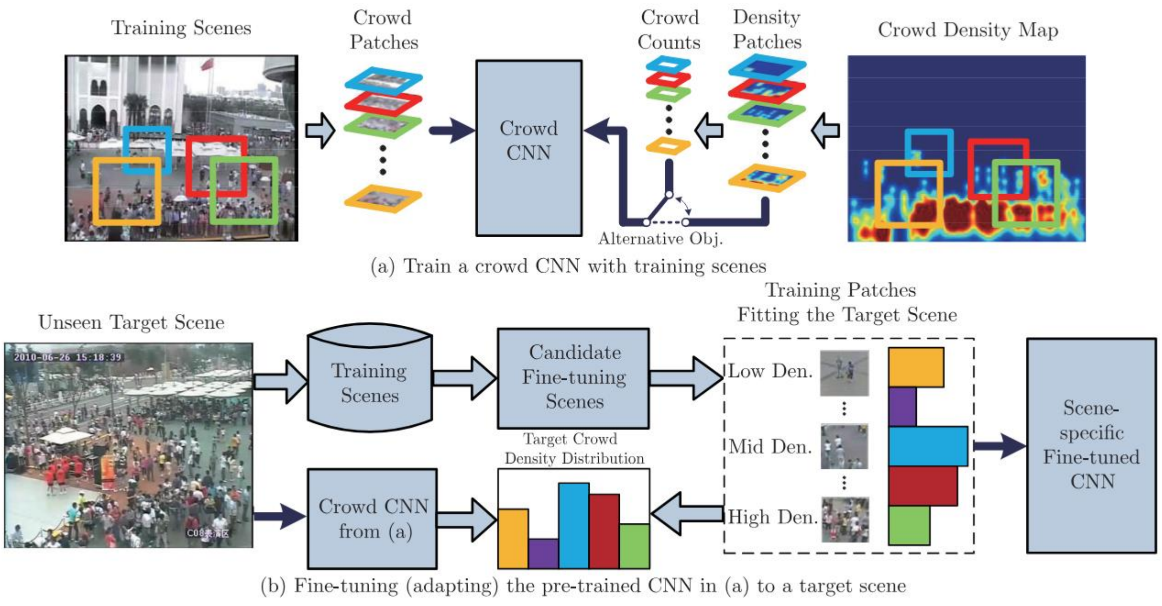

Training Scenes
Crowd Patches
Crowd CNN
Crowd Counts
Density Patches
Crowd Density Map
Alternative Obj.
(a) Train a crowd CNN with training scenes
Unseen Target Scene
Training Scenes
Candidate Fine-tuning Scenes
Target Crowd Density Distribution
Training Patches Fitting the Target Scene
Low Den.
Mid Den.
High Den.
Scene-specific Fine-tuned CNN
Crowd CNN from (a)
(b) Fine-tuning (adapting) the pre-trained CNN in (a) to a target scene

Figure 3: The proposed model / training method in paper [3]. Firstly, a CNN is trained with the training scenes (a). Then it is fine-tuned for a specific target scene (b).

Paper [4] explores Expanded Context Aware Network (ECAN). The model dynamically uses multi-scale contextual information with varying field sizes shown in Figure 4. This strength allows the method to handle perspective distortions and different crowd densities effectively. It is also highly efficient. However, the weakness of such a method means that it is highly complex and being a relatively new practice its effectiveness on unseen data is yet to be seen. This method builds off CSRNet models by adding a context aware mechanism into the model. In return this builds an incredibly powerful model for counting crowds by not only counting people but by also calculating the amount based on the amount of space and perspective distortion.

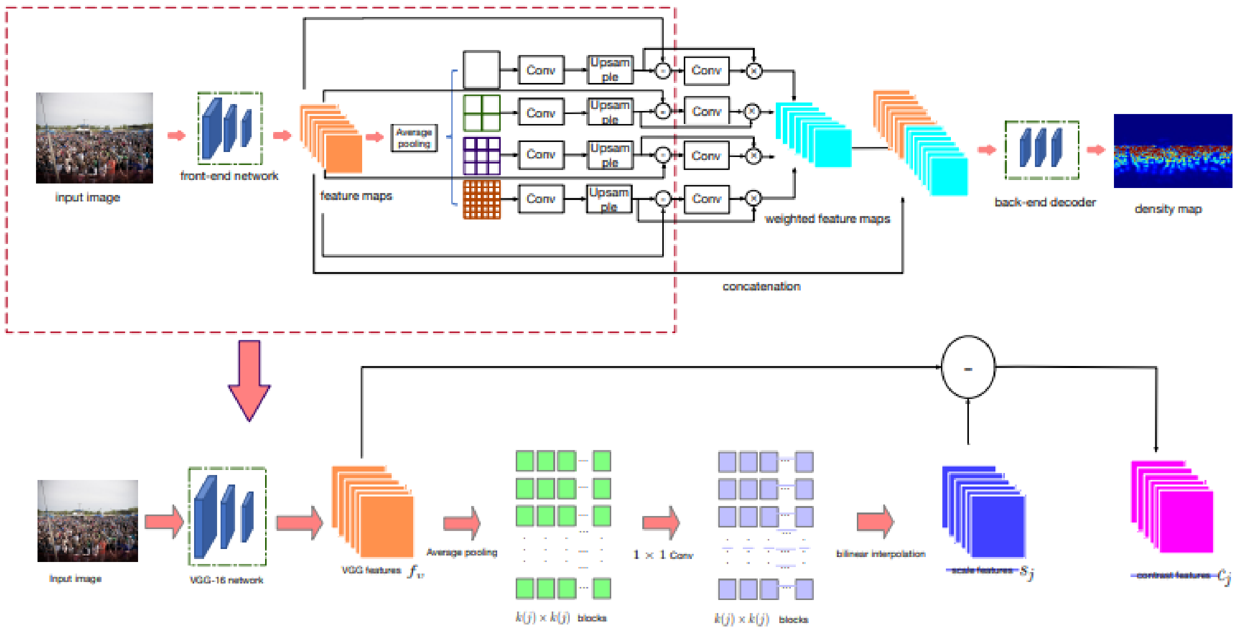


Figure 4: Content Aware Network. (Top) The entire CAN network comprised of feature mapping and the context aware that result in a density map. (Bottom) This section is an expanded version of the initial context feature mapping.

Paper [5] explores PSL-Net. This model uses pseudo square labels as training supervision for crowd counting and localisation. This model predicts the probability of center points within reasonable grids, which increases detection of individuals in dense crowds. Figure 5 is an example of this method in action. The strengths of the model are also its anchor-free approach which simplifies the detection process and increases flexibility. It also has the precise label assignment which introduces a many-to-one matching algorithm which enhances the model's accuracy. The downside is that it is still quite a complex model. Because of these strengths it makes it an excellent model for crowd counting.

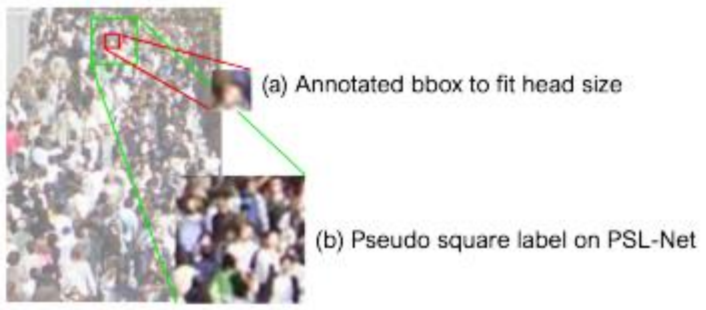
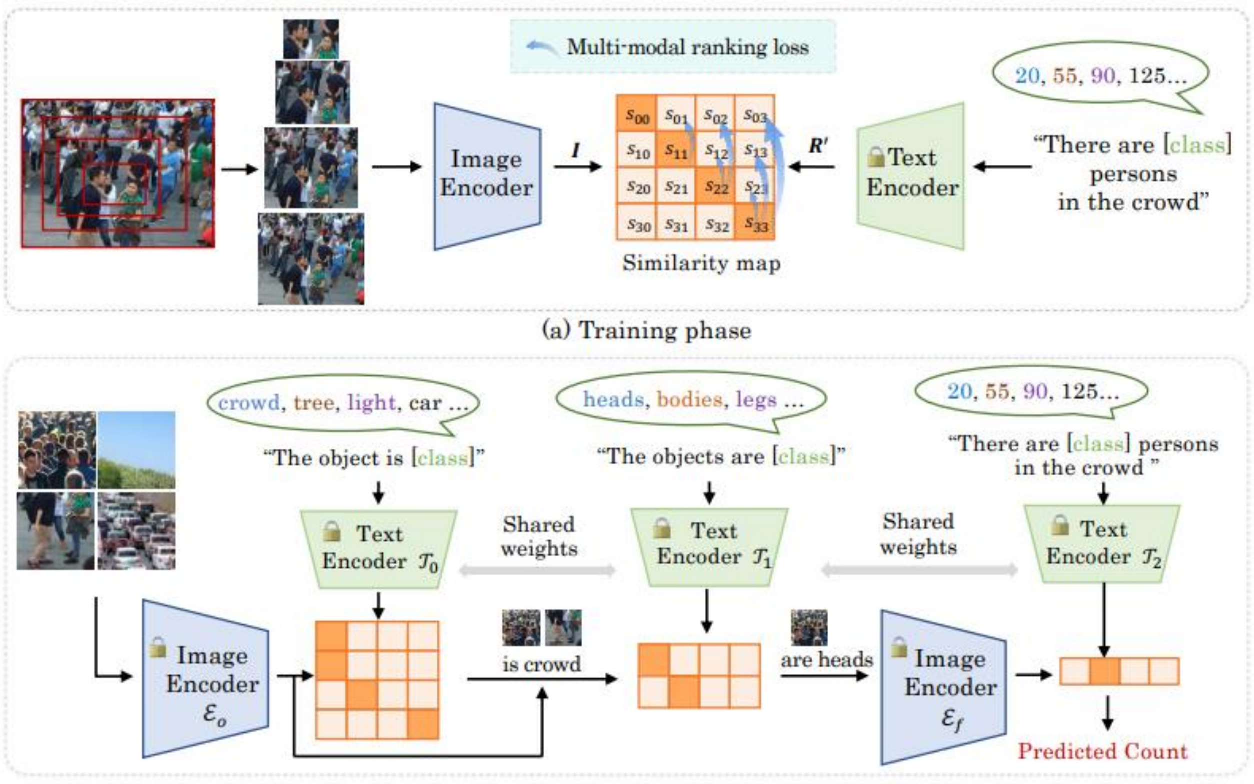

(a) Annotated bbox to fit head size
(b) Pseudo square label on PSL-Net

Figure 5: PSL-Net Algorithm's example approach.

Paper [6] explores the CrowdCLIP algorithm which is a version of the MCNN method. This method is an unsupervised crowd counting method using the CLIP vision language model. It uses ranking text prompts to tune the image encoder and a filtering strategy to map crowd patches to count intervals. These filters are based on the objects that are visible in the image and allow for the model to put them in individual classes, this is shown in Figure 6. CrowdCLIP also outperforms most unsupervised and some supervised methods when it comes to crowd counting. This method is also built off the idea of MCNN combined with the visual aid of CLIP. This is what makes it such a brilliant method as it builds off one of the more successful methods that is filled with strengths and not many weaknesses.



Multi-modal ranking loss
20, 55, 90, 125...
Image Encoder
I
R'
Text Encoder
"There are [class] persons in the crowd"
s_{00} s_{01} s_{02} s_{03} s_{10} s_{11} s_{12} s_{13} s_{20} s_{21} s_{22} s_{23} s_{30} s_{31} s_{32} s_{33}
Similarity map
(a) Training phase
crowd, tree, light, car ...
heads, bodies, legs ...
20, 55, 90, 125...
"The object is [class]"
"The objects are [class]"
"There are [class] persons in the crowd"
Text Encoder $\mathcal{T}_0$
Shared weights
Text Encoder $\mathcal{T}_1$
Shared weights
Text Encoder $\mathcal{T}_2$
Image Encoder $\mathcal{E}_o$
is crowd
are heads
Image Encoder $\mathcal{E}_f$
Predicted Count

Figure 6: An overview of CrowdCLIP's methodology in the encoding of images and progressive object filtering.

Paper [7] was inspired by the success of R-CNN methods. This paper instead uses the CNN as a feature extraction and uses the cascade Adaboost algorithm to obtain the head regions as pre-processing for the CNN. The final classification is then done by a linear SVM. This approach greatly reduced classification times, achieves good performance, and even outperforms baseline methods.

Paper [8] proposes a regression guided detection network for crowd counting. With aims to improve robustness of detection-based approaches for small heads by leveraging density maps. It achieves the best performance for RGB-D counting and localisation and demonstrates the effectiveness of density maps to the performance of crowd counting methodologies.

Paper [9] takes advantage of bi-knowledge transfer to reliably count crowds across visual domains. They use a labelled source domain to discover bi-knowledge transfer between regression and detection-based models. The dual source knowledge is complementary as they capture different modalities. They do this by formulating mutual transformations between the two outputs, the regression and detection model as two scene-agnostic transformers. This enables the knowledge transfer between the two. Analysis shows that the method is advantageous against the most recent cross-domain crowd counting methods and further demonstrates the effectiveness of multi-metric models for crowd counting.

Paper [10] developed a method that can accurately estimate the crowd count from an image with arbitrary perspective and crowd density. They proposed an MCNN to map the image to its crowd density map. The features learnt by each column CNN are adaptive to changes in head size due to perspective shift or image resolution. They do this by implementing filters with receptive fields of different sizes. To assess the effectiveness of this method, they had to collect and label new, more challenging, data. Not only does their analysis shows that the method outperforms all existing methods, but also that it can be easily transferred to a new dataset once trained on one.

3 Data

We chose two datasets for this project, the first is a Crowd Counting set found in [12], the second dataset was ShanghaiTech found in [11].

The first dataset contained image frames from a video recorded from a camera in a mall. This camera was stationary, so the only differences between the frames in the set were the crowd counts. The data set contained 2000 annotated video frames, with a dot label on each pedestrian in the image. The images were originally of size 640x480. The dataset was split into a training set with 80% of the samples (1600) and a testing set containing 20% (400 samples).

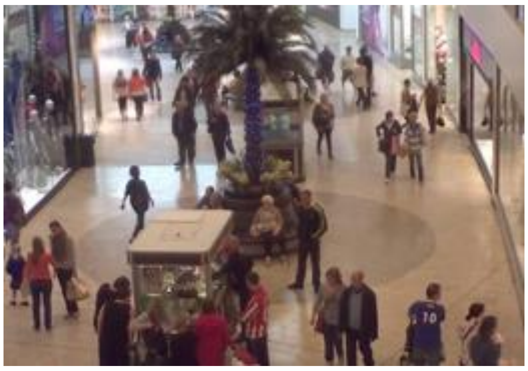


Figure 7: Kaggle dataset sample training image.

The second dataset contained 1198 annotated crowd images, split into two parts. Part-A contained 482 images collected from the internet and had higher crowd densities on average and Part-B contained 716 images collected on the streets of Shanghai. This dataset was pre-split in test and training splits: Part-A had 300 images for training and 182 images for testing; Part-B had 400 images for training and 316 for testing. This dataset contained a much wider range of environments, camera angles, and noise in its data than the Kaggle set.

Figure 8: Shanghai Part A sample image (left), Shanghai Part B sample image (right).

The datasets were resized to 156x156 to reduce computational complexity, though a larger resolution would have benefited the models. These two datasets were chosen to allow the comparison and evaluation of each of the chosen algorithms in different environments.

4 Methodology

4.1 Poisson Regression

Existing approaches for crowd counting have explored the use of Gabor filters or PCA as feature extraction for use in deep learning models but not for regression approaches, and not in combination. This paper aims to address the shortcomings of the previous approaches by implementing both a Gabor filter and PCA for feature extraction for use in a Poisson regression network.

A Gabor filter is a linear filter used for texture analysis, which is particularly effective in capturing spatial and frequency characteristics of images. It was proposed by Dennis Gabor and is inspired by the visual cortex of mammals [14]. The Gabor filter's response is determined by its orientation and frequency, making it highly suitable for edge detection and texture representation. The mathematical representation of a Gabor filter is a sinusoidal wave modulated by a Gaussian envelope.

With the local spatial and frequency information representation of the images, we can apply PCA to reduce its dimensionality, while retaining the essential information. PCA will be very effective for this task as regression models tend to struggle with high-dimensional input data – whether it be due to computational complexity, the curse of dimensionality, or simply overfitting. The Gabor filtered data is then vectorised, as PCA requires. It was also then standardised, to ensure PCA considers each component equally.

There were many hyper-parameters to fine-tune with this approach. For the Gabor filter and Poisson regression model, we used a grid search to find the optimal parameters. For a full breakdown of the searched parameters, please see Appendix B. For PCA, the only parameter for us to tune was the number of components to retain after the dimension reduction. We chose to retain the number of components that explained 95% of the variance in the data. So, the number of components retained was different for each of the datasets, seen in Table 1.

Table 1: Number of PCA components retained for each dataset.

	Kaggle	**Shanghai A**	**Shanghai B**
PCA Components	1238	260	349

To perform a grid search on the Gabor filters parameters, we developed a custom transformer and scoring function. The developed scoring function included the training of a Poisson regression model which meant that it also had to include PCA, and therefore vectorisation and standardisation of the Gabor filtered data. The regression model was trained with the current iterations Gabor filter applied. This increased the computational complexity of the grid search considerably, but it was the most appropriate way to score each Gabor filter estimator. This increased complexity meant reducing the number of parameters in the search, which may lead to limitations in the performance of this method.

Table 2: Optimal Gabor filter parameters found for each dataset.

	ksize	**sigma**	**theta**	**lambda**	**gamma**	**psi**
Kaggle	(5, 5)	1	$\frac{\pi}{2}$	5	1	$\frac{\pi}{2}$

Shanghai A	(5, 5)	5	$\frac{\pi}{2}$	5	1	0
Shanghai B	(31, 5)	1	$\frac{\pi}{2}$	10	0.5	$\frac{\pi}{2}$

The Kaggle optimal parameters collectively configure the Gabor filter to be a localised, vertically oriented edge detector that captures medium-scale textures with a circular Gaussian envelope and a phase shift. The optimal Shanghai A parameters allow for detecting vertical edges and medium-scale textures in an image, with a broader Gaussian envelope for more global feature detection within a localised region. The optimal Shanghai B filter parameters are useful for detecting elongated vertical features and coarser textures in an image, with a narrow Gaussian envelope focusing on localised vertical details. The elliptical shape due to $gamma = 0.5$ makes it particularly suited for enhancing vertical edges and textures.

The Poisson regression grid search was much less computationally expensive, as it had a standard scoring method. The optimal parameters for each dataset are shown below in Table 3:

Table 3: Optimal Poisson regression model parameters for each dataset.

	Alpha	**Fit intercept**	**Max Iterations**	**Solver**	**Tolerance**
Kaggle	300	True	100	‘lbfgs’	0.1
Shanghai A	20	True	1000	‘lbfgs’	0.01
Shanghai B	20	True	500	‘lbfgs’	0.01

Both parts of the Shanghai dataset shared optimal regularisation (alpha) and tolerance values. The regularisation value implies a relatively strong regularisation, and given the large variation in these datasets, this makes sense. The tolerance value of 0.01 is fairly moderate, meaning the models are iterating until the change in the loss function is less than 0.01, which should lead to a fairly precise model. The Kaggle model had an even larger regularisation value, and this worked to prevent overfitting that was observed through manual testing of this parameter. This model also had a tolerance higher than the Shanghai sets. A possible explanation for this is that the Kaggle set has much less variation in its data, as the setting is static, and so the model is reaching convergence quite quickly. This higher tolerance value was also validated through manual testing.

4.2 U-Net

The only form of U-Net previously used was a W-Net in which the goal was to estimate a density map rather than a crowd count. A W-Net is a reinforced U-Net, it does this by attaching two U-Nets together, hence the name. This makes W-Nets exceptional at image segmentation tasks but too large in theory for purely crowd counting tasks. This paper will aim to explore using a small typical U-Net and its effectiveness in crowd counting.

U-Nets were originally developed to create a high performing network for medical imaging tasks. U-Nets earned their name through their design architecture; consisting of a series of encoder layers that concatenate into decoder layers later within the model. The encoding layers allow the model to learn the features of the image through a series of convolutional and max pooling layers. Whereas the decoding layers combines the features learned in the encoders to reproduce/restore the image. As mentioned earlier it gets these features through the concatenation layers or “skip connections”. U-Nets are symmetrical about the number of encoder and decoder stages. U-Nets also excel in working with small datasets.

A U-Net design was chosen as it was believed that it would be effective in crowd counting due to its high-level feature extraction and its small size / computational requirements. Although in theory for crowd counting tasks the decoder stage of a U-Net is not necessary as all the information necessary for the task will be captured within the encoders. However, decoders will still be implemented to observe the learned features after reconstruction for analysis purposes. The final proposed model can be observed in the figure below.

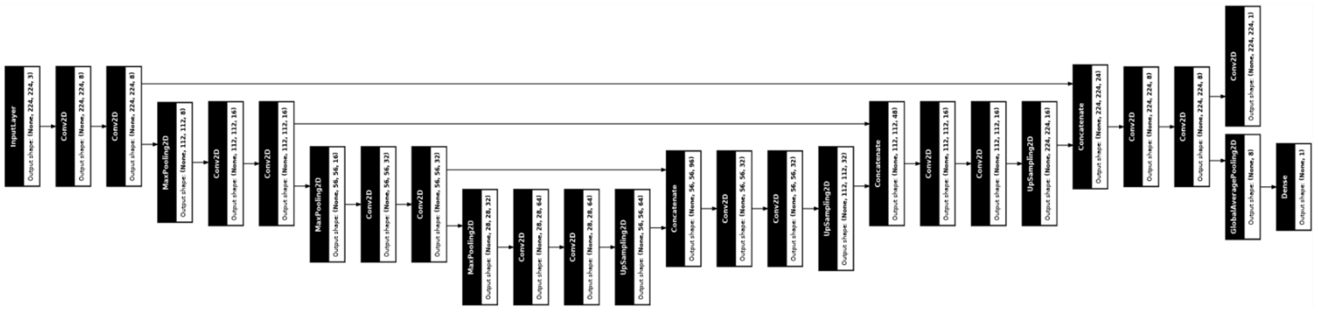


Figure 9: The proposed U-Net model architecture for crowd counting. The model has approximately 0.12M parameters. There are two outputs the decoded image (ideally a density map however unlikely) and the crowd count.

As seen above the proposed model has 3 encoder and decoder stages with the appropriate skip connections. The model has approximately 0.12 million parameters and produces two outputs, the decoded image and the crowd count. The decoded image is ideally a density map however it is unrealistic as the model is purely training towards the crowd count. It's also unrealistic as the desired or target density map is unknown, and the model is being fed dummy density maps (images of the same size as the original with one channel all 0's) such that the output can be observed. However, it should hopefully provide some insight into the model although it is completely unnecessary and can be skipped overall.

The proposed model will be trained with as high of a batch size as computationally possible for (224, 224, 3) image inputs and as many epochs until the observed validation loss creates a callback with a patience of 5 epochs. This should train a sufficient model for crowd counting with a chance at some insight into the model through the decoded image.

4.3 SANet

Scale Aggregation Network (SANet) was built off the back of many different CNN methods, MCNN being a considerable influence, and aims to estimate density maps for a result of a precise crowd-count over an efficient count. SANet finds features at multiple scales, enhancing its ability to process various crowd densities. This paper will explore Scale Aggregation Networks and its effectiveness in crowd-counting.

SANet was originally developed for the specific purpose of crowd-counting. Its name comes from its unique design architecture which aggregates features at different scales. The network consists of multiple convolutional layers that are processed at various scales. The multi-scale approach allows both fine-grained and coarse features to be learned making it highly effective at crowd estimation.

The core innovation of SANet lies in its multi-column back-end network. This component includes several convolutional columns, each with a different receptive field, enabling the extraction of features at multiple scales. This multi-scale approach ensures that both local and global features are captured, which is crucial for accurate crowd density estimation.

The final component of SANet is the density map estimator, which consists of several convolutional layers. These layers process the aggregated multi-scale features to produce a density map that accurately represents the crowd count in the input image.

SANet was chosen due to its high-level feature extraction that has proven its capability in dense crowds. However, the model has only been trained and tested on high density crowds so the training method will have to be modified to correctly process low density crowds. The original training method is seen below in Figure 10.

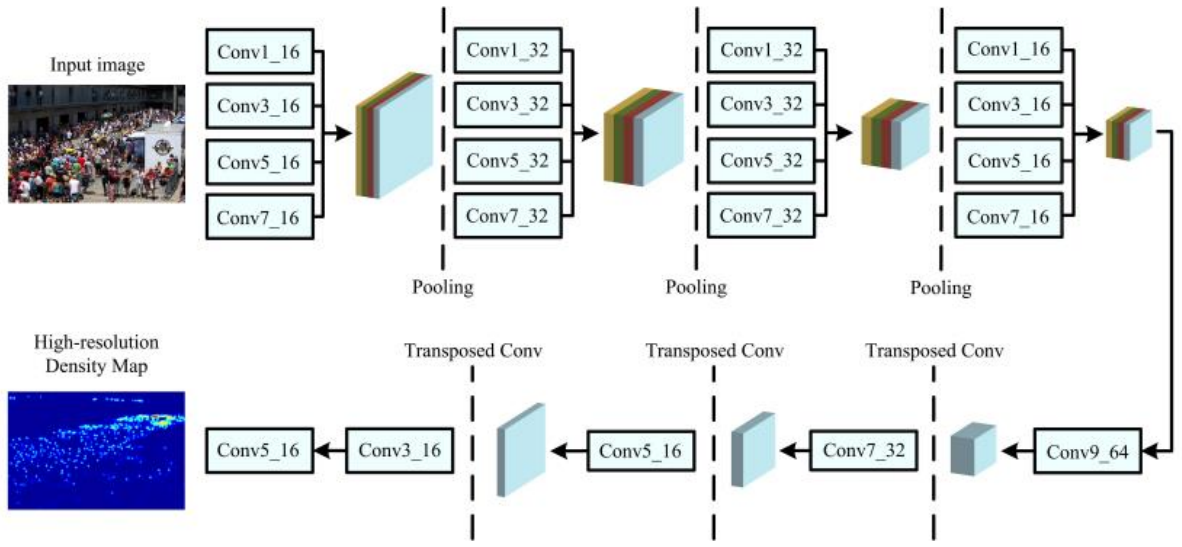


Figure 10: The Original Architecture of SANet. [13]

The more simplified version will focus more on some of the larger features finding a middle ground between being effective for high- and low-density crowds. This proposed model will be trained on up to 50 epochs with a patience of 3 epochs. This will allow for the model to accurately calculate all the different scales that it needs to process effectively. This model also contains many parameters therefore it will take a long time to train. However, this is the aim of the model as it focuses more on accuracy over effectiveness.

5 Evaluation

5.1 Poisson Regression

In total there were three models trained for crowd counting with Poisson regression. One for the Kaggle dataset, and one for each of the Shanghai datasets.

The model trained on the Kaggle dataset performed quite well. It achieved an R^2 of 0.98, which indicates that the model was fitting very well to the data. It scored a mean squared error of 1.20 on the training set and 7.37 on the testing set. This implies the model was overfitting slightly but was within an acceptable range. Figure 11 shows the models predicted counts against the true counts. This performance was unexpected, but maybe shouldn't have been; the static setting and larger sample size of the Kaggle dataset were both extremely advantageous and were most definitely a factor in its relatively high performance.

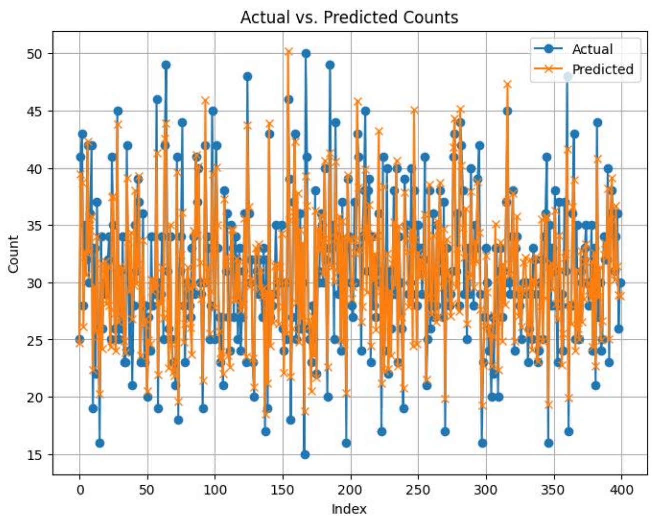


Figure 11: Poisson regression model trained on Kaggle dataset, showing good performance on the testing set with an MSE of 7.37.

The two Shanghai sets had similar R^2 scores to the Kaggle model, both with 0.96. Again, implying the model explains a large proportion of the variance in the target variable. But in the case of the two Shanghai models, that was not accurate. As seen in the below Figure 12, both models were performing extremely poorly. This implies the R^2 value was inflated, and that the models were fitting to noise in the training sets, and subsequently failing to generalise to unseen data.

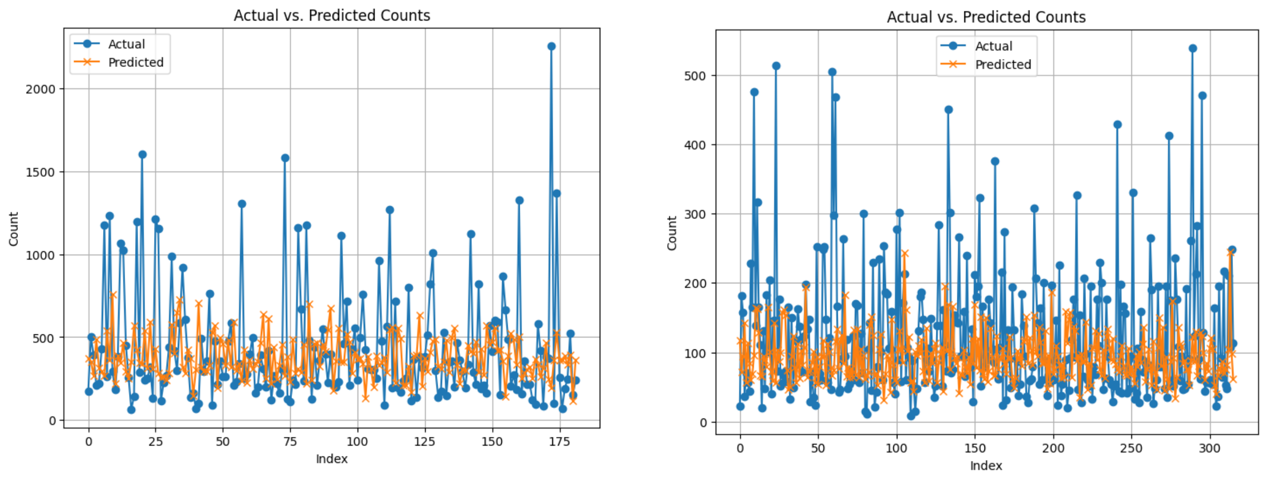


Figure 12: Shanghai A (left) and Shanghai B (right) models showing poor performance on the testing set. With MSE scores of 145,906 and 10,952, respectively.

The Shanghai A model had an MSE of 9,098 on the training set and 145,906 on the testing set. The Shanghai B model achieved an MSE of 335 on the training set and 10,952 on the testing set. The Shanghai B model did outperform the A model, but both struggled considerably.

There were two major factors impacting the performance of the models: the average crowd density, and the static setting of the Kaggle dataset versus the dynamic setting of the Shanghai datasets. As seen by the increase in MSE as the average crowd density increases; the Kaggle dataset with the smallest crowd density, Shanghai B with an intermediate density, and Shanghai A with the highest density, it has a massive impact on the models ability to generalise. This factor is only exacerbated by the lower number of samples in both Shanghai sets and their dynamic environments. Not only does the model have a more challenging task with higher crowd densities, but it has to generalise to changing environments too. For this reason, the model requires much more data, as well as higher image resolutions in hopes of a more desirable performance.

As regression is a non-deep learning method, the training and inference times were quite minimal. All models, including the Gabor filtering and PCA, took less than four seconds to train, with the exception being the Gabor filter and PCA for the Kaggle set which took 24 seconds in total. And each had inference times even shorter at 0.001 seconds or less. Overall, very negligible times. The strengths of these times do not, however, make up for the lack of performance on the Shanghai datasets, both of which provide no applicable use. The model trained on the Kaggle set, though, is very promising, and with further tuning of Gabor filter parameters (and more compute time/power), could rival the performance of the deep learning methods to be detailed below.

The approach using Gabor filters, PCA, and Poisson regression demonstrates promise but ultimately falls short in practical applications. While it performs well in controlled, static settings, it struggles with the complexity and variability of real-world scenes. For this approach to truly rival leading-edge deep learning methods in dynamic environments and applications, further enhancements in feature extraction, model complexity, and adaptability are necessary.

5.2 U-Net

The U-Net like the other models was trained on three datasets, the Kaggle dataset and part A and B of the Shanghai dataset.

The U-Net model performs extremely well on the Kaggle dataset. The model achieves a mean squared error of approximately 4.2 on the testing set. An example of the output of the model can be seen in the figure below, where the input and the expected output are compared to the predicted count and the decoded image. The low MSE can also be seen in the figure where the predicted count lies within +-3 of the actual count.

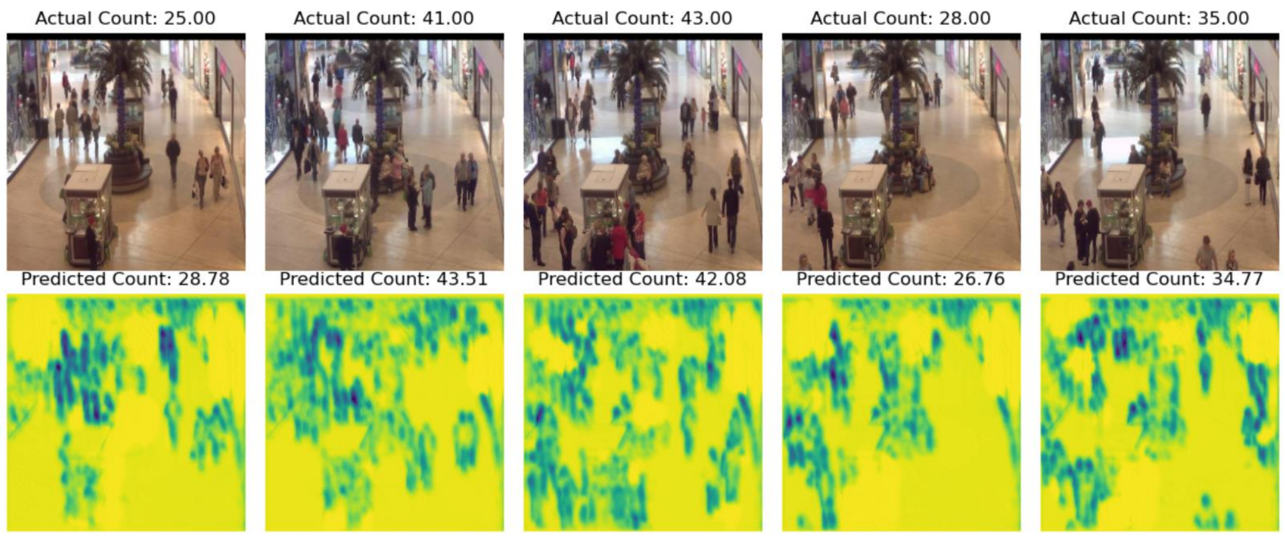


Figure 13: Example output of the Kaggle model. Both the predicted count and decoded image (bottom row). The top row is the input image and the actual count.

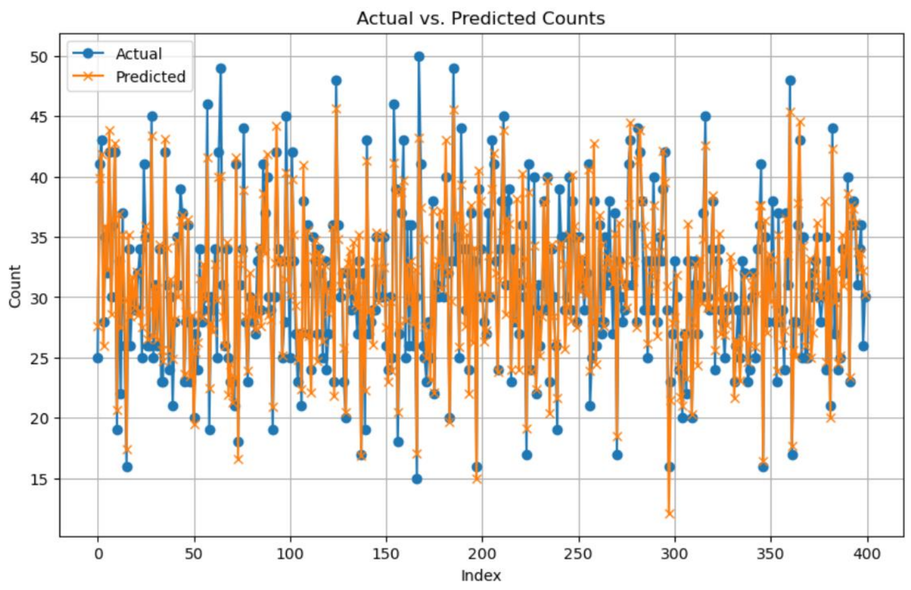


Figure 14: The performance of the Kaggle model in terms of the actual vs predicted for the test set. The model struggles to accurately predict when the count is 45+.

The model takes about 5 minutes time to train on the training set and takes about 4 seconds to predict the entire testing set. It can also be observed in the actual vs predicted counts graph that the model struggles to correctly identify when there are towards 50 people in the image. The long train time is due to the high number of samples within the Kaggle dataset but is also what makes the model so accurate. Another reason the model performs so well is due to the static location of the images allowing the model to "learn" the environment and distinguish from people. This can be somewhat seen in the decoded image output where people are generally highlighted or outlined in blue. This is not a crowd density map but more of a person location map / this is a person.

Unlike the Kaggle dataset the Shanghai images were not all taken in the same location so the environment cannot be learned. This means that the model has to focus more on learning person detection than environmental detection. The model trained on part A of the dataset had an MSE of approximately 129090, whilst the model trained on part B had an MSE of approximately 9209. The resulting outputs can be seen below.

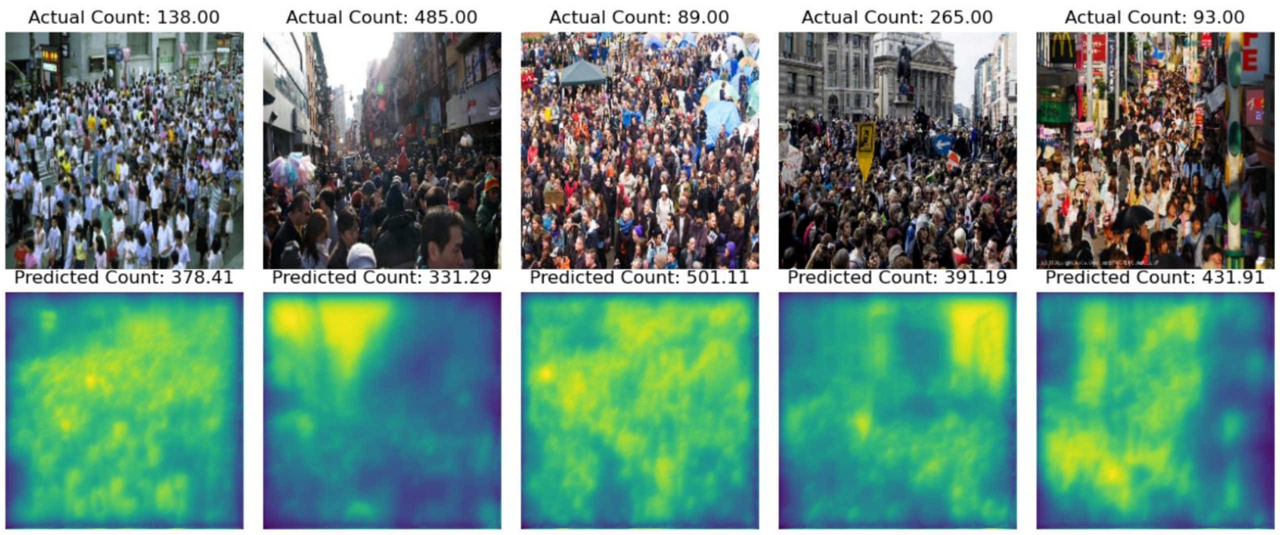


Figure 15: Example output of the Shanghai Part A model. Both the predicted count and decoded image (bottom row). The top row is the input image and the actual count.

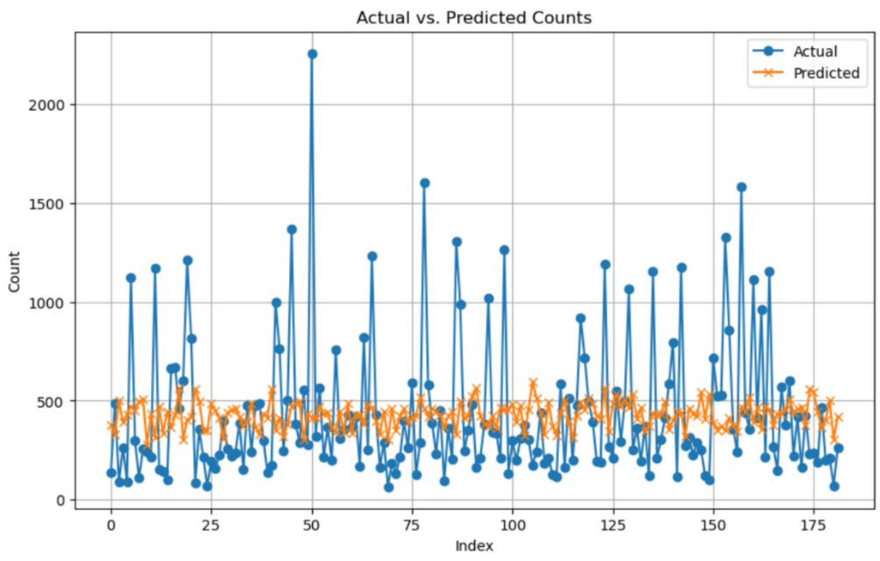


Figure 16: The actual vs predicted count for the Shanghai part A model. With a mean squared error of 129090.

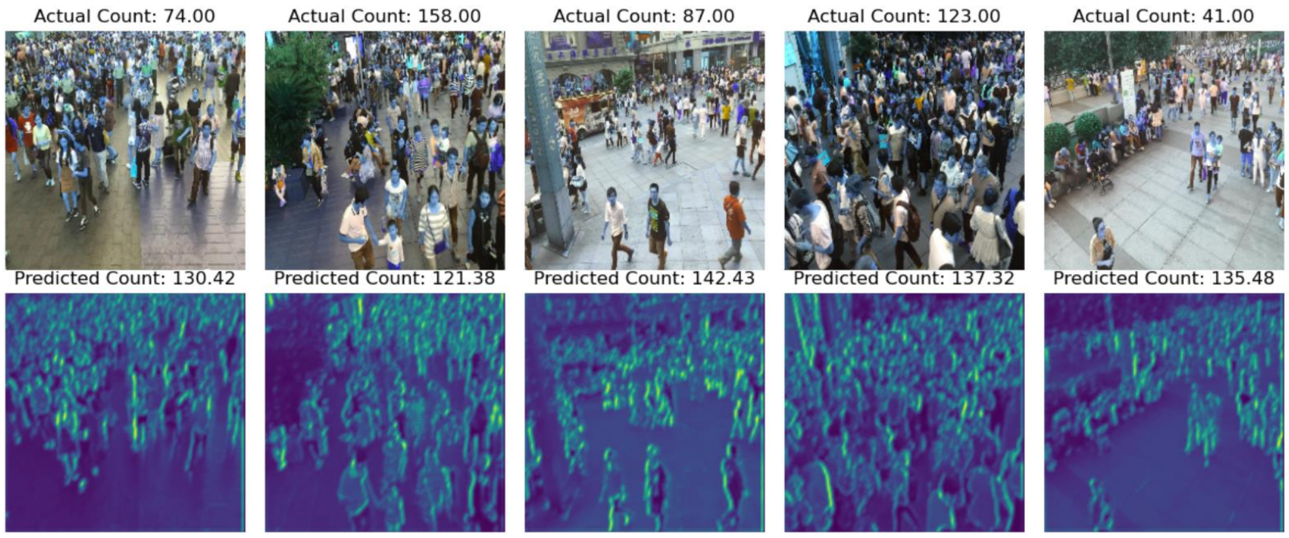


Figure 17: Example output of the Shanghai Part B model. Both the predicted count and decoded image (bottom row). The top row is the input image and the actual count.

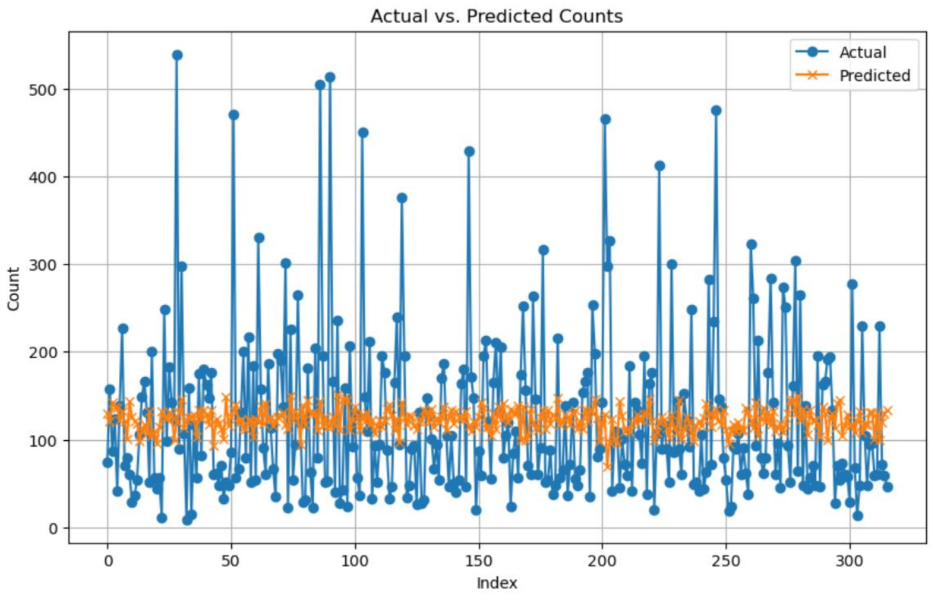


Figure 18: The actual vs predicted count for the Shanghai part B model. With a mean squared error of 9209.

As observed in the figures above, the model performs significantly worse on the Shanghai sets than the Kaggle set. This is for two main reasons, firstly as mentioned earlier the camera location is no longer static, secondly the low number of training samples. Unlike the Kaggle set, both Shanghai datasets have significantly less samples to train on, part A with 300 and part B with 400 samples. This is also why the model trained on part B performs better than the model trained on part A. However, a third argument can be made that its due to the difference in range between the two models, the part A goes up to a count of 2000 and the part B only goes to 500.

The model seems to learn more of an average from the training data then the number of people in the image. The model takes significantly less time to train on the Shanghai sets due to the low number of samples with an average training time of one to two minutes.

For the model to actually output a crowd density map, it's not ideal to use a U-Net, as you would want to keep down sampling the image as seen in the first paper under related work. Alternatively, an advanced loss equation could be developed which takes the predicted count into account and maps accordingly. Both are ideas for future work in the model.

5.3 SANet

The SANet was trained on the three data sets, the Kaggle mall data set, as well as A and B of the Shanghai dataset.

Overall, the model performed well on both the Kaggle dataset and the Shanghai dataset. Scoring a low MSE score for all three datasets. Firstly, we will talk about the Kaggle dataset.

The Kaggle dataset model was a large dataset that was set in the same spot every time; therefore, the model was able to learn from its environment quite quickly. The first few epochs had a mean squared error of approximately 14.8 to begin with as shown on the bottom of Figure 19. This high error count was due to the count of the model being initially structured for high density crowds. However, a restructuring of the model by taking out a low-level density layer allowed for the training to run faster allowing for more of training without it being too long. This decreased the MSE to approximately 3.9 as shown in the top of Figure 19.

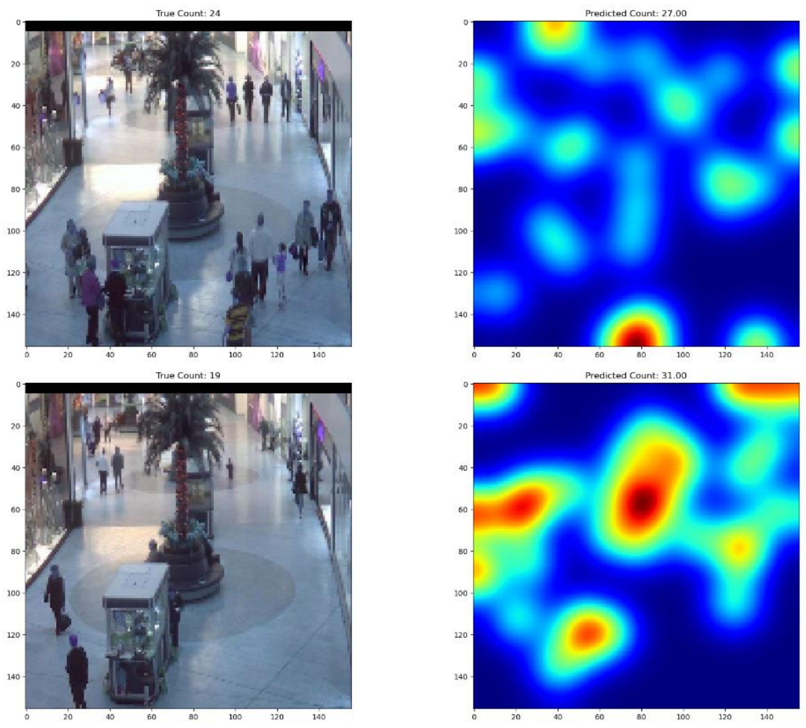


Figure 19: Final Training Results (Top), Bottom Model Initial Training Results (Bottom).

This dataset is quite large however, boasting a set of 2000 images in the same location. This put the SANet model at a disadvantage as it takes a long time to model in comparison to other models. The other disadvantage is that the model trains using density maps of tight crowds. Therefore, dispersed crowds confuse the model as seen in figure 20. The density map is chaotic and does not work as effectively. The Poisson regression model and U-net model mentioned early in the paper trained and tested for a few seconds and 5 minutes respectively, while on the other hand this model trained for 10 minutes and took 10 seconds to evaluate. This is much slower than both models and is slower than most other popular crowd counting models like MCNN which take a similar time to the U-net model. This clearly shows that in this circumstance where a quick and accurate count is needed in a low dense area, either a faster DNN method or a non DNN method is recommended over SANet.

As a test, however, we also trained it for a maximum of 30 minutes allowing for the lowest MSE with an epoch patience of 5. This resulted in a model that had an MSE of 1.24, this conveys the possibility of the model being more accurate when given the time, however, this length of time is not plausible for a real situation.

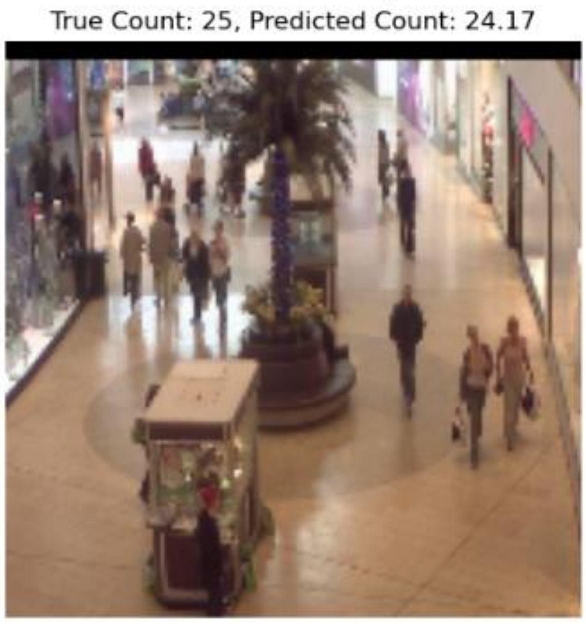


Figure 20: Results of SANet trained for 30 minutes.

The next datasets that the model was trained for were Shanghai Tech A and B. This high density, data set is where the SANet performed its best and was what it was designed to do. Since the environment was no longer a factor, as every image was a new image, the multi-scaled density maps can work effectively to find an accurate crowd count. The Shanghai Tech dataset was trained much longer than the Mall Data set as there were a lot more features that needed to be captured to create an accurate crowd count. The dataset was trained for a total of 60 minutes each and resulted in an MSE of 1024 for part A and 938 for part B. Even though the dataset is smaller, the MSE is naturally going to be much larger as there are a lot more people in each image.

One of the effects of the redesigned model structure that we created, is that it is a lot more accurate, in reference to percentage correct, to whatever the main type of crowd is. For example, for the Mall dataset, the dataset was sparse groups of people, so it was able to closely predict sparse groups. However, with the Shanghai dataset, since most scenes are densely populated crowds, the scenes like Figure 21, were heavily over predicted. This is because it believes that most crowds in this data set are at a minimum, a few dozen people, where sometimes they are only a couple people.

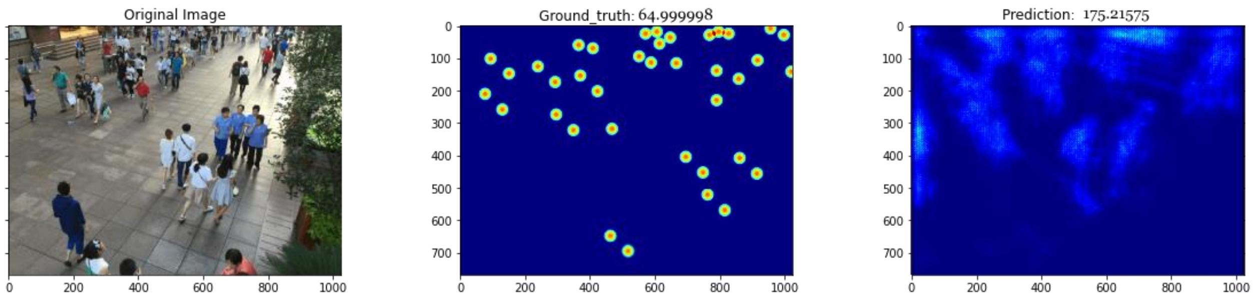


Figure 21: Example of a sparse crowd and the substandard performance from the model.

The data that SANet tested the best with, was when the count of people was 200 or greater. This is thanks to the fact that the various stages of pooling allowing varied sizes of crowds to be pooled correctly. From there the model can accurately predict the count of the group based of the size and the density of the crowd. As seen below in Figure 22, the predicted count gets closer to the actual count when focusing on the percentage of correct counted.

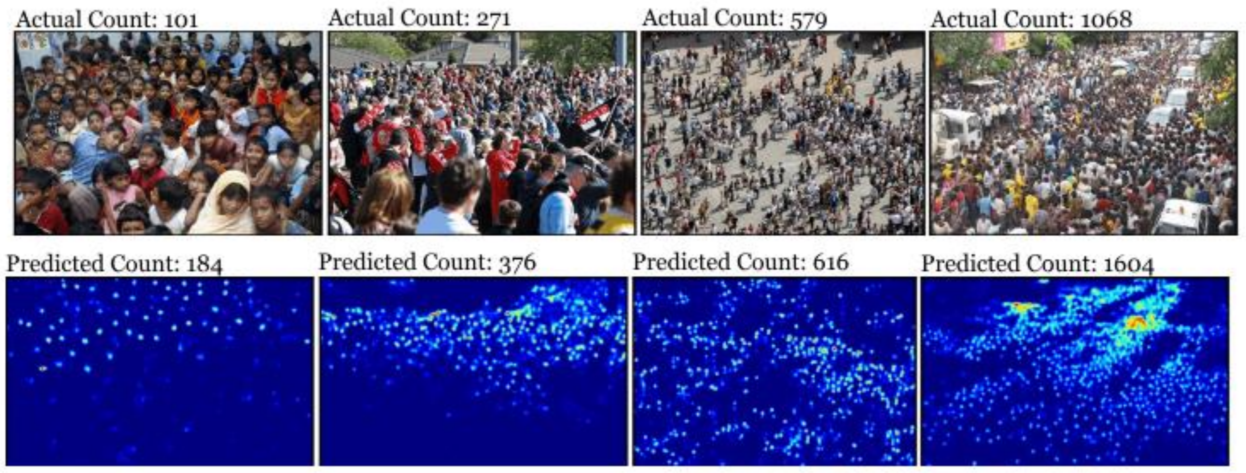


Figure 22: Example Output of Shanghai Tech Data Set. Actual count and Original (Top Row) Predicted count and density map (Bottom Row).

When looking at the total results of Shanghai Part B in comparison to the finalised model, Figure X, you can clearly see that throughout the model it closely predicted the count. This is because of the thorough training it went through. It is evident that the prediction did poorly under a count of 100 and overestimated the count of people in general. When comparing SANet to the U-net and Poisson regression models, it does outperform but at the heavy cost of efficiency. The time it takes to train the other models are 6 to 150 times faster than SANet, respectively. When choosing an appropriate method in practice, you would have to focus on what the goals of your model are if it is accuracy or efficiency. Once trained, however, the inference time was quite low being 50ms per image. To generate a highly accurate SANet model you would need to train the model for many hours with a high-powered GPU, which has been performed to get an MSE of 107.25 [13].

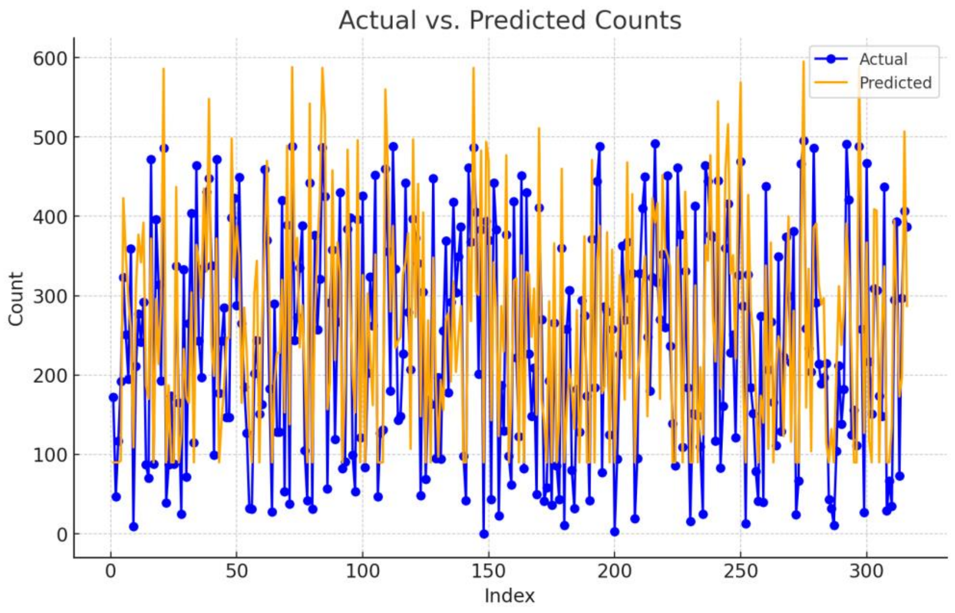


Figure 23: The actual vs predicted count for the Shanghai Part-B model. With a mean squared error of 1024.

6 Conclusion and Future Work

In this study, we set out to assess the performance of three distinct machine learning algorithms in crowd counting and their ability to adapt to dynamic environments. All three algorithms performed quite well on the Kaggle images, but that was somewhat expected given that it was the least challenging of the datasets. The SANet method performed the best of the three, with an MSE of just over 1. It did take substantially longer to train though, as did the U-Net model, than the Poisson regression method. But given that only happens once, it is not a deciding factor. The inference times are much more impactful to the model's applicability, and each method achieved a similar time of less than 50ms. The Poisson regression approach had the benefit of being simpler in terms of network architecture and design but depending on the extent of the grid search on Gabor parameters, can be computationally demanding. It also had a very minimal training time but was otherwise outclassed by both deep-learning approaches.

The effectiveness of each approach on the ShanghaiTech dataset follows a similar pattern to the Kaggle dataset. Albeit, with a much higher discrepancy in accuracy. The SANet model performed the best, with an MSE of 1024 on Part A, and 938 on Part B. The other two methods had MSE's above 100,000 and 9,000 on Part A and B, respectively. The SANet model did again train for much longer than the other approaches but it certainly makes up for it with its performance. There is still some room for improvement with this model, most of which can be solved with more training time, but it could already provide a lot of use in real applications.

There are possibilities for future work with all three methods. The Poisson regression approach could be improved by implementing a deep network in the feature extraction pipeline. Or the regression model itself could be replaced by a DCNN with PCA and Gabor filtering as feature extraction. Both of which would help to mitigate some of the shortcomings previously identified with this approach. With the U-net method, it could implement larger filter sizes or more encoder/decoder stages to help it learn more complex representations in the data. However, this would possibly make the model less lightweight adding parameters. At that point it may be better to switch to a model purely with encoder layers such as that in paper [1]. Additionally, the U-net method could implement a loss function for the decoded image which would change it to a density map. When it comes to improving the SANet method, there are quite a few ways to improve it, but all are more computationally demanding. The first and easiest method would be to re-introduce the removed pooling layer and then train the model for a longer period. This would allow the model to learn slightly more complex relationships in the data, so it would be able to calculate the crowd density and crowd count more accurately. The second way that the SANet model could be improved, would be to train the model with a combination of Euclidian loss as well as local pattern loss as seen in paper [13]. These in combination allow for the fine tuning of the model to narrow down an accurate crowd count and has been proven effective, as in paper [13].

References

[1] X. Shi, X. Li, C. Wu, S. Kong, J. Yang, and L. He, "A Real-Time Deep Network for Crowd Counting," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020, pp. 2328-2332. DOI: 10.1109/ICASSP40776.2020.9053125

[2] L. Boominathan, S. S. S. Kruthiventi, and R. V. Babu, "CrowdNet: A Deep Convolutional Network for Dense Crowd Counting," in *Proceedings of the 2016 ACM on Multimedia Conference (MM '16)*, Amsterdam, Netherlands, Oct. 2016, pp. 640-644. DOI: 10.1145/2964284.2967300.

[3] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene Crowd Counting via Deep Convolutional Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 833-841.

[4] W. Liu, M. Salzmann and P. Fua, "Context-Aware Crowd Counting," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 5094-5103, doi: 10.1109/CVPR.2019.00524.

[5] J. Ryu and K. Song, "Crowd Counting and Individual Localization Using Pseudo Square Label," in IEEE Access, vol. 12, pp. 68160-68170, 2024, doi: 10.1109

[6] D. Liang, J. Xie, Z. Zou, X. Ye, W. Xu and X. Bai, "CrowdCLIP: Unsupervised Crowd Counting via Vision-Language Model," 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 2023, pp. 2893-2903, doi: 10.1109/CVPR52729.2023.00283.

[7] C. Gao, P. Li, Y. Zhang, J. Liu, and L. Wang, “People counting based on head detection combining Adaboost and CNN in crowded surveillance environment,” *Neurocomputing*, vol. 208, pp. 108–116, Oct. 2016, doi: https://doi.org/10.1016/j.neucom.2016.01.097.

[8] D. Lian, J. Li, J. Zheng, W. Luo, and S. Gao, “Density Map Regression Guided Detection Network for RGB-D Crowd Counting and Localization,” Jun. 2019, doi: https://doi.org/10.1109/cvpr.2019.00192.

[9] Y. Liu, Z. Wang, M. Shi, Shin’ichi Satoh, Q. Zhao, and H. Yang, “Discovering regression-detection bi-knowledge transfer for unsupervised cross-domain crowd counting,” *Neurocomputing*, vol. 494, pp. 418–431, Jul. 2022, doi: https://doi.org/10.1016/j.neucom.2022.04.107.

[10] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, “Single-Image Crowd Counting via Multi-Column Convolutional Neural Network,” *IEEE Xplore*, 2016. https://ieeexplore.ieee.org/document/7780439

[11] “Papers with Code - ShanghaiTech Dataset,” *paperswithcode.com*. https://paperswithcode.com/dataset/shanghaitech

[12] F. Mena, “Crowd Counting,” www.kaggle.com. https://www.kaggle.com/datasets/fmena14/crowd-counting/data (accessed May 31, 2024).

[13] Cao, X., Wang, Z., Zhao, Y., Su, F., & Sun, X. (2018). Scale Aggregation Network for Accurate and Efficient Crowd Counting. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 734-750).

[14] Wikipedia Contributors, “Gabor filter,” *Wikipedia*, Feb. 12, 2019. https://en.wikipedia.org/wiki/Gabor_filter

Appendix

Appendix A: Contributions

Name	Student Num	Tasks	%	Signature
Jacob Woods	n10234853	• Poisson Regression Methodology & Evaluation	33.33	[illegible]
Flynn Kassulke	n10809511	• SANet Methodology & Evaluation	33.33	[illegible]
Jean Terblans	n11246332	• U-Net Methodology & Evaluation	33.33	[illegible]
Shared Work		• Introduction • Data Section • Conclusion and Future Work • Related Work		

Appendix B: Gridsearch Values

Gabor Filter

ksize_height	5, 11, 31, 51
ksize_width	5, 11, 31, 51
sigma	1.0, 3.0, 5.0
theta	0, pi/4, pi/2
lambd	5.0, 10.0, 15.0
gamma	0.5, 1.0
psi	0, pi/4, pi/2

Poisson Regression Models

alpha	0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 15, 20, 50, 100, 150, 200, 300, 350, 400, 1000
fit_intercept	True, False
solver	'lbfgs', 'newton-cholesky'
max_iter	10, 100, 200, 300, 500, 750, 1000, 1500, 200
tol	0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1