Final Report on Crowd Counting Model

Adil Ismagambetov
School of Engineering and Digital Sciences
Nazarbayev University
Astana, Kazakhstan
adil.ismagambetov@nu.edu.kz

Ryan Goh
School of Computing and Information Systems
Singapore Management University
Singapore
ryan.goh.2022@scis.smu.edu.sg

Project Repository on GitHub
Video Presentation Link

***Abstract*—This report presents the development and evaluation of deep learning-based crowd counting models, with a focus on ResNet-backboned CNNs and CSRNet. After reviewing several methods—detection-based, regression-based, and CNN-based—we selected CNNs for their effectiveness in handling high-density and variably distributed crowds. The UCF-QNRF dataset was used due to its diversity and volume. Preprocessing steps included random cropping, padding, and noise injection to simulate real-world conditions. Density maps were generated as labels for model training. We experimented with ResNet-18 and ResNet-34 backbones, tuning learning rates to achieve optimal performance, with the best model (ResNet-34, 1e-5 lr) achieving an MAE of 7.56. CSRNet was also implemented and trained using Colab GPUs, showing strong performance with an MAE of 9.24 under optimal batch and epoch configurations. The results demonstrate that both models can achieve competitive performance on challenging crowd counting tasks, with trade-offs between computational cost and accuracy. Future work includes integrating dataset fusion, optimizing hyperparameters, and expanding model generalization.**

I. Introduction

Crowd Counting is a problem that involves the estimation of the exact number of people captured in a certain area [10]. In terms of computer vision, this problem focuses on crowd counting in a single frame or in a sequence of frames taken from a video file. Since the progress report, we have started building crowd-counting models with ResNet18 backbone, and we have tested CSRNet model. This Final report presents the whole pipeline of this project and show the results in the end.

II. Related Works

From the papers we reviewed, the common angles devised to tackle the crowd-counting problem are as follows [12]:

- **Convolutional Neural Network (CNN)**: There exist many methods (multi-scale fusion [16], attention-based models [13], etc.) which all boil down to the use of a neural network to learn and count objects in crowds.
- **Detection Based Methods**: These methods extract the defining features of the entire image that could be used to identify the objects (edges, textures, Haar wavelets, etc) and then use classifier algorithms (SVM, Random Forest) to detect or classify objects. Works well for most cases when objects are sparse across the image, but performance is greatly affected as density of objects increase.
- **Regression Based Methods**: Further divisible into individual-based and density map-based methods. Individual-based regression, according to the example usage of it by Ke et al. [1], uses the extracted and normalised foreground to learn and obtain multiple regressions, producing a final object count for the image. Density map-based regression, proposed by Lempitsky and Zisserman [7], reflects the spatial distribution information of the crowd and further increases counting accuracy.

After a literature review on existing crowd-counting models at the beginning of the semester, we decided to use a CNN approach. It performs better than other crowd-counting models, handles uneven distribution of people in a frame and works with a variety of background scenes [8] [10]. By using a CNN, we hoped to produce a competent model that can estimate crowd numbers from an image accurately regardless of background noise.

III. Dataset Description

At the beginning, to expand on our model's versatility, we considered the use of multiple datasets based on the following criteria: ease of use for our training/testing purposes, diversity of data, and the amount of storage space the dataset occupies.

For the dataset mentioned in our proposal, UCF-QNRF consists of 1535 images scraped from Flickr, Web Search and Hajj footage with headcounts ranging from 50 to 12,000 per image. It is divided into train and test sets of 1201 and 334 images respectively. In addition, the images are taken from different countries, which will hopefully increase the training and testing quality of the model [4].

We have also considered other commonly used datasets related to crowd counting mentioned by other academic works. These include the UCF_CC_50 dataset and ShanghaiTech dataset [17]. The UCF_CC_50 dataset has a low number of images and a large variance in headcount, which could provide an additional challenge to our model if this dataset is used as a test set [18]. The ShanghaiTech dataset consists of 2 parts: part A has images crawled from the Internet and is split into train and test subsets in the ratio of 300:182 images respectively, and part B was meticulously captured by the authors of the cited paper on the busy Shanghai streets, and

has been split into train and test subsets in the ratio 400:316 images respectively. The ShanghaiTech dataset also boasts a unique viewpoint for every image, which has the potential to provide better training quality for our model [17].

Datasets shown here use a variety of different annotation methods [6]:

- **Point/Dot annotations**: Each object detected will be marked with a dot typically on the centre of the object's head. It is typically used to estimate the density of objects present in an image, but summing up the dots allows it to also be used for count-regression models to estimate the total count in the image.
- **Bounding boxes**: A rectangular box is drawn over the detected object to encase as much of the object/the head of the object as much as possible.
- **Total count annotations**: used in detection-based or regression-based counting approaches, all the objects of interest are counted. The grand sum of objects present will then be assigned to the image.

Aside from the datasets mentioned above, here are all the datasets considered, including datasets that were not explicitly mentioned [5], [6], [15]:

TABLE I
CROWD COUNTING DATASETS CONSIDERED

Dataset	# of Images	Headcount/Img	Resolution
UCF-QNRF [2]	1535	50 - 12,865	Varies
ShanghaiTech-A [17]	482	33 - 3,139	Varies
ShanghaiTech-B [17]	716	9 - 578	768×1024
UCF_CC_50 [3]	50	94 - 4,543	Varies
WorldExpo10 [14]	3,980	1 - 220	576×720
DroneRGBT [11]	3,600	1 - 403	512×640

Ultimately, we concluded the UCF-QNRF dataset has a good mix of what we were looking for, providing a solid baseline for preparing and evaluating our model's performance. In the progress report we have mentioned that we may combine some of the above mentioned datasets for better accuracy. However, once we did the preprocessing and augmentation parts for UCF-QNRF, we had 4804 images for training. Adding more images on top would require better GPU or much more time, which we didn't have at the moment. Therefore, in the end we used only UCF-QNRF dataset.

IV. DATA PREPROCESSING

UCF-QNRF dataset contains 1201 images and annotations for the training set. However, they all have a different shape. At first, we decided to scale all of them to a target size of 1024X1024. Unfortunately, this alters the aspect ratio of the images, which would make it difficult for the model to work with real images that have the usual-for-eye aspect ratio. Then we decided to take 4 random crops from every image, where the shape of random crop is a target size. In the cases where the image was smaller than the target size, we applied padding to it. These crops were saved in "Train_resized" folder. With random cropping, we increased the number of training images from 1201 to 4804. We also decided to use Data Augmentation to make our training data more diverse. There are multiple ways to do that, starting from flipping and rotating images and ending with adding noise/brightness. It was decided to use noise injection on cropped images. Our model has a high chance to encounter this type of data, because noise injection would simulate old-outdated or dysfunctional cameras. We randomly added noise to half of the pictures, where the type of noise (gaussian, salt-pepper, and poisson) was also randomly selected. To optimize the code, noise injection was applied during random cropping.

Testing set contained 334 images. It was decided to split it to Validation and Test sets, where each of them would have 167 original images. These images were also cropped to a target size, which increased their number to 668 for validation and 668 for the test set. These images were placed in "Test_resized" and "Validation_resized" folders.

After that, since we decided to use Convolutional Neural Networks, we needed density maps as our label for images. Density maps for each image were created using annotations. These density maps will be labels for our Neural Network. Basically we will input an image to a model, it will process it and output its density map. Then, we will sum all the bright points in density map, and this sum will be estimation of number of people.

Before progress report, we were taking random crops of size 1024 by 1024, which we realized was quite heavy for our laptops. Then, we decided compromise and took crops of size 224 by 224, which is also input size for ResNet models.

In the end, our dataset is 4804 training images with noise injection, and 668 images for both validation and testing. All the images of size 224 by 224.

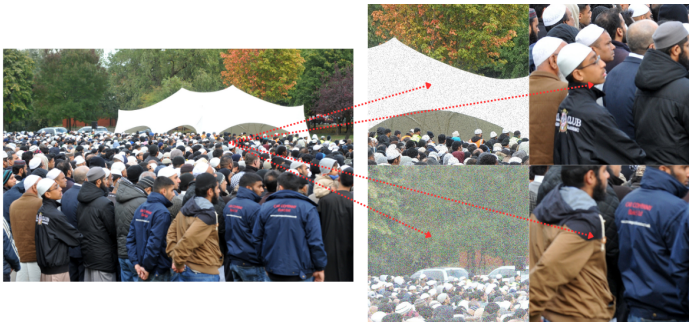

Fig. 1. Example of Random Cropping and Noise Injection.

V. RESNET BACKBONES SOLUTION

We started with 2 possible solutions. The first is to implement the CSRNet crowd-counting model [9], and the other was to create the crowd-counting model using ResNet as the backbone. Ryan was working with the CSRNet model, while Adil worked with the ResNet ones.

For a ResNet solution, we started with the lighest model, ResNet-18. At first, we were getting close to 0 solutions. It was explained by the fact that values in density maps were very small, so CNN model was making its weigths either 0 or

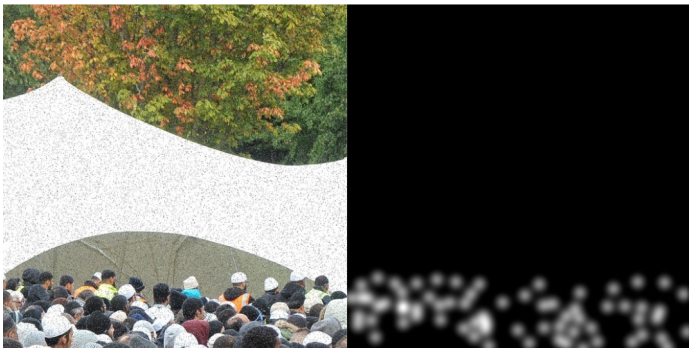

Fig. 2. Training Image and its Density Map.

close to zero. To fix this issue, we just multiplied the density maps by a scale factor of 100 before comparing it with the output of the model. Therefore, we will have a higher loss, which will result in higher weights. However, the sum of the output will be also multiplied by this scale factor. To prevent this, we were just dividing the output of CNN by this scale factor.

With a ResNet-18 backbone, we were training for 50 epochs and saving the best model in terms of validation loss. We have tried it with different learning rates (1e-3, 1e-4, 1e-5, and 1e-6). We used Kaiming He initialization, Adam optimizer with weight decay of 1e-5 and Mean Squared Error as a loss function. The architecture of CNN model is as follows: ResNet-18 architecture up to classification layer in addition with a series of transposed convolutional layers to upsample feature map to match the input size (224, 224). During training for 50 epochs, we were saving the best model based on its validation loss. All training and validation loss curves are present in the GitHub repository. We then have calculated the mean absolute error for the test data set for all the models. To clarify, the mean crowd count of the test dataset is **29.4**, with a maximum of 301 and a minimum of 0.

TABLE II
MAE FOR DIFFERENT LEARNING RATES USING RESNET-18 BACKBONE

Model	Mean Absolute Error (MAE)
ResNet-18 1e-3 lr	11.38
ResNet-18 1e-4 lr	10.06
ResNet-18 1e-5 lr	**8.54**
ResNet-18 1e-6 lr	10.78

We have repeated the same procedure with ResNet-34 backbone.

TABLE III
MAE FOR DIFFERENT LEARNING RATES USING RESNET-34 BACKBONE

Model	Mean Absolute Error (MAE)
ResNet-34 1e-3 lr	13.02
ResNet-34 1e-4 lr	8.51
ResNet-34 1e-5 lr	**7.56**
ResNet-34 1e-6 lr	8.55

As we can see from the results, the best performing ResNet backboned solution turned out to be ResNet-34 backboned model with learning rate of 1e-5, Kaiming He initialization, Adam optimizer, and weight-decay of 1e-5.

Fig. 3. Training and Validation loss Curves for Best Performing ResNet backboned Model.

We have reached the best mean absolute error of 7.56 for test dataset at epoch 30. We also can clearly see that after that it clearly started overfitting. All the 8 models (ResNet-18 and ResNet-34) are saved in the GitHub repository.

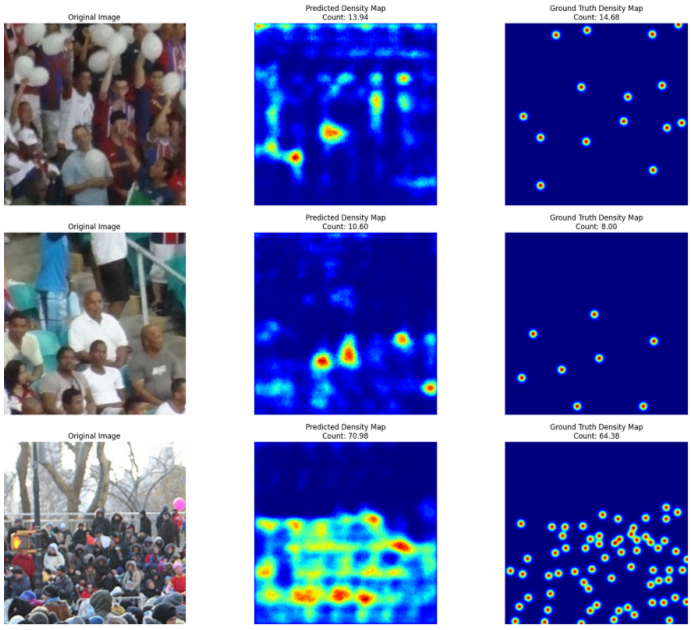

Fig. 4. Some outputs for test dataset for best performing ResNet-34 model.

VI. CSRNET SOLUTION

We also delved into using a crowd counting-centric, pre-trained model called CSRNet. As explained in the previous report under the "Future Work" section, initial research suggests that it would have required too much computational power and time to train and test. However, we found it to be feasible to train it with the T4 GPU compute provided by Google Colab and hence proceeded with creating a full pipeline, consisting of pre-processing, training, and testing, with automated results recording steps.

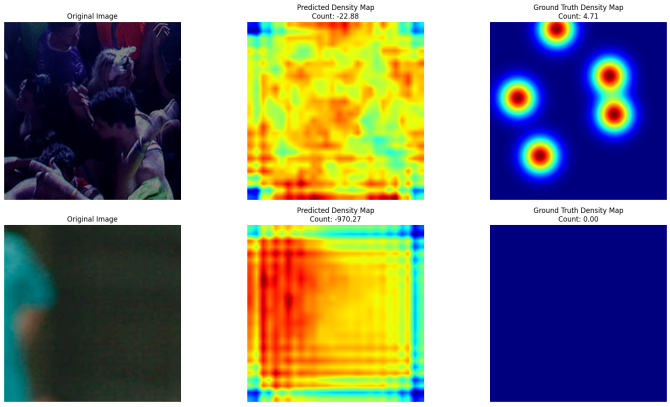

Fig. 5. Faulty training pipeline produced inaccurate model predictions

We set up 2 machines (Ryan's laptop and Google Colab platform) to increase the rate of pipeline completions, which required us to make some minor adjustments to the directory references and device type that Pytorch was configured to. When the daily GPU quota had exceeded for one Google account, another version of the Colab pipeline on another account had to be created since referencing the image and density data had to be done differently.

A. Challenges

In the early stages, we were unsure if the drastically low training loss (around 1×10^{-6} or lower) was due to the pre-trained model's sophistication or errors in the pre-processing/training step. After some trial prediction runs, it was evident that it was the latter due to largely inaccurate predictions and suspicious artifacts produced from the prediction density maps (blue corners in Fig 5). We managed to find these errors:

- Due to our efforts to save on computational cost in training and to augment the dataset, we decided to re-crop our photos into standardized dimensions, which also helped with data augmentation. However, this produced many cropped images with little to no people in it, heavily impacting training quality. This was easily solved by setting a conditional statement to only allow crops with 5 or more people in it.
- Training density maps were not saved in the correct format (jpg instead of the proper png format), which made density map information "invisible" and useless to model training.

After fixing these errors, we still found that the training loss between the ResNet and CSRNet models were still disproportionately different ($20-30$ vs 1×10^{-4} in first epoch), so we came to the conclusion that it was due to the crowd countring-centric nature of CSRNet that it produced much lower training losses.

B. Testing Methodology

Before proceeding with testing our trained models, we did a control experiment to ensure our training had an impact on

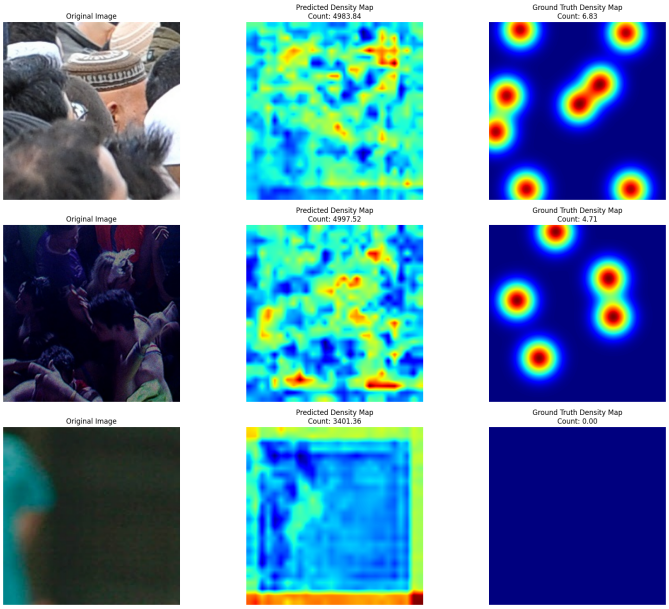

Fig. 6. Baseline predictions from pre-trained, untouched CSRNet model

prediction performance. Figure 6 shows We manually tested different hyperparameters, of which are listed below:

- **Training and Validation Batch Size**: This varies the number of images loaded to the model during the training and validation phases. Changing this will impact training times in exchange for stability of gradient estimates and model generalizability.
- **Epochs**: Due to our use of the pre-trained model, we had to be careful with overfitting the models.
- **Model Learning Rate**: We had to balance this with the time constraints and model performance in mind.

We felt that these were the hyperparamaters that impacted the performance the most, and other parameters (optimizer learning rate, etc) were not too significant. Although we could have done automated parameter optimization, the time taken to create the code and computational burden to run the code is not justifiable for this project. However, this will be used as an option if the opportunity arises.

C. Results

TABLE IV
MEAN ABSOLUTE ERROR FOR TOP 5 CSRNET MODELS PRODUCED

Train & Vali. Batch Size	Epochs	Learn. Rate	MAE
10	3	1e-5	9.24
30	3	1e-5	14.73
7	3	1e-5	89.81
20	3	1e-5	100.84
30	2	1e-5	125.52

Table IV shows the results obtained. We have saved all the model weights trained and MAE values, along with some of the visualizations produced. All these resources are available on the GitHub repository, linked in the conclusion.

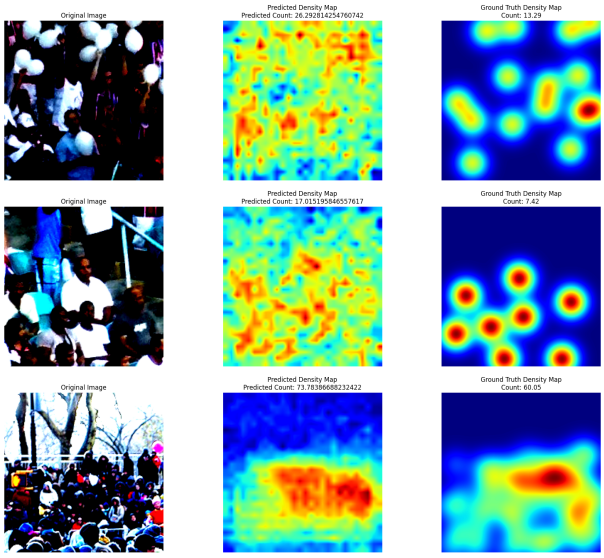

Fig. 7. Predictions from the best CSRNet Model

D. Discussion

Figure 7 shows some visualizations we did for the best CSRNet model's predictions. In this figure, model prediction counts were about +/- 5-20 people off, which is a big improvement from the start of the project. We found that keeping the batch sizes between 10 to 30 images helped with extracting the most out of the model's training quality. Training epochs beyond 3 given the batch sizes resulted in overfitting, all in part of the model's pre-trained nature. Due to CSRNet's heavy VGG-16 backbone (front-end) and dilated convolutions (back-end), increasing the number of epochs while reducing batch size would not have been time-feasible.

VII. Conclusion

In this project, we explored deep learning techniques for solving the crowd counting problem by implementing and evaluating CNN-based models, particularly ResNet-based architectures and CSRNet. The use of the UCF-QNRF dataset allowed us to train models on a wide range of crowd densities and image conditions, and the data preprocessing pipeline—including random cropping, padding, and noise augmentation—contributed to the robustness of our models. It is also worth mentioning that density maps for both models differed. For ResNet-34 model we used standard deviation of Gaussian filter of 3 when computing density maps. For CSRNet this parameter was 15, which is why density dots appear larger for CSRNet.

The ResNet-34 based model with a learning rate of 1e-5 yielded the best performance with a mean absolute error (MAE) of 7.56, demonstrating the importance of both model selection and hyperparameter tuning. CSRNet, though computationally heavier, achieved a close MAE of 9.24 when trained with optimal settings using Google Colab's T4 GPU. Despite initial difficulties with data formatting and cropping strategy, we were able to refine our pipeline and significantly improve prediction accuracy.

Both models showed promising results in counting accuracy and spatial density estimation. The trade-off between computational efficiency (ResNet) and architectural specialization (CSRNet) highlights the practical considerations in deploying such systems. In future work, we aim to explore more advanced augmentation strategies, incorporate multiple datasets for better generalization, and potentially integrate lightweight transformer-based or attention-enhanced architectures to push the accuracy even further without incurring excessive training costs.

All models, training logs, and visualizations have been made publicly available in the GitHub repository linked with this report. GitHub repository: https://github.com/Adil7777/CrowdCounting

References

[1] Ke Chen, Chen Change Loy, Shaogang Gong, and Tao Xiang. Feature mining for localised crowd counting. 01 2012.

[2] H. Idrees, M. Tayyab, K. Athrey, D. Zhang, S. Al-Maddeed, N. Rajpoot, and M. Shah. Composition loss for counting, density map estimation and localization in dense crowds. September 8-14 2018.

[3] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. pages 2547–2554, 2013.

[4] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-ma'adeed, Nasir Rajpoot, and Mubarak Shah. *Composition Loss for Counting, Density Map Estimation and Localization in Dense Crowds: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*, pages 544–559. 09 2018.

[5] Akbar Khan, Kushsairy Abdul Kadir, Jawad Ali Shah, Waleed Albattah, Muhammad Saeed, Haidawati Nasir, Megat Norulazmi Megat Mohamed Noor, and Muhammad Haris Kaka Khel. A deep learning approach for crowd counting in highly congested scene. *Computers, Materials and Continua*, 73(3):5825–5844, 2022.

[6] Muhammad Asif Khan, Hamid Menouar, and Ridha Hamila. Revisiting crowd counting: State-of-the-art, trends, and future perspectives. *Image and Vision Computing*, 129:104597, 2023.

[7] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[8] Bo Li, Hongbo Huang, Ang Zhang, Peiwen Liu, and Cheng Liu. Approaches on crowd counting and density estimation: a review. *Pattern Analysis and Applications*, 24:853–874, 2021.

[9] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1091–1100, 2018.

[10] Akshita Patwal, Manoj Diwakar, Vikas Tripathi, and Prabhishek Singh. Crowd counting analysis using deep learning: A critical review. *Procedia Computer Science*, 218:2448–2458, 2023.

[11] Tao Peng, Qing Li, and Pengfei Zhu. Rgb-t crowd counting from drone: A benchmark and mmccn network. In Hiroshi Ishikawa, Cheng-Lin Liu, Tomas Pajdla, and Jianbo Shi, editors, *Computer Vision – ACCV 2020*, pages 497–513, Cham, 2021. Springer International Publishing.

[12] Nitin Kumar Saini and Ranjana Sharma. Deep learning approaches for crowd density estimation: A review. In *2023 12th International Conference on System Modeling Advancement in Research Trends (SMART)*, pages 83–88, 2023.

[13] Rahul Rama Varior, Bing Shuai, Joseph Tighe, and Davide Modolo. Multi-scale attention network for crowd counting, 2019.

[14] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. pages 833–841, 06 2015.

[15] Lu Zhang, Miaojing Shi, and Qiaobo Chen. Crowd counting via scale-adaptive convolutional neural network. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1113–1121, 2017.

[16] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. pages 589–597, 06 2016.

[17] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. pages 589–597, 2016.

[18] Zhen Zhao, Miaojing Shi, Xiaoxiao Zhao, and Li Li. Active crowd counting with limited supervision. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 565–581, Cham, 2020. Springer International Publishing.

VIII. Appendix

The division of work is as follows:

- **Adil**: Optimisation of ResNet training/testing pipeline to reduce training time, and enhanced ResNet model performance. Did relevant part of report.
- **Ryan**: Optimising/Refactoring of code in CSRNet training/testing pipeline. Did further research into other papers on methods that employ similar methods as us, exploration on other pre-built models that could improve performance. Did relevant part of the report.