BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS464 - Final Report - Group 23

Berk Güler, Munib Emre Sevilgen,
Şeyma Aybüke Ertekin, Yağmur Özkök

Project Name: Crowd-Counting Using Convolutional Neural Networks

Table of Contents

1. Introduction

In this project, we have worked on the crowd counting problem. Crowd counting is essentially generating an estimation of people in an image, which generally contains many people hence the crowd in the name. We have worked with a single data set that consists of two parts, that is available on Kaggle and is offered by ShanghaiTech [1]. Later, we have used the data augmentation method to increase our data during training.

We have trained two different convolutional neural network models on our data, namely CSRNet [2] and MCNN [3].

We have obtained better results with CSRNet compared to MCNN. However, data augmentation worked well for MCNN, while we cannot take the response we expected from CSRNet models.

2. Description

Crowd counting is a machine learning technique used to estimate the number of people in a crowd in an image. Crowd counting techniques can be used to estimate and analyze attendance rate in the events, to provide an estimate about how much resource or how many staffs needed in the business or it can be used for video surveillance or managing the traffic. In the face of Covid-19, it can also serve as a very useful tool by reporting crowded areas and thus preventing people from getting together in large groups.

Crowd counting may become a difficult problem, especially, if the scenes dealt with contain dense crowds. When it comes to sparse crowds, it is possible to count people by detection. However, if the crowd is dense, density estimation methods are required in order to count people on a scene. Density estimation methods require using convolutional neural networks.

In our project we have worked with two well-known CNN models for crowd counting. We have tried to use data augmentation methods and observe the effects of some minor changes on parameters. Basically, we dealt with the data required and the effect of the data on the results.

3. Methods

3.1. Dataset

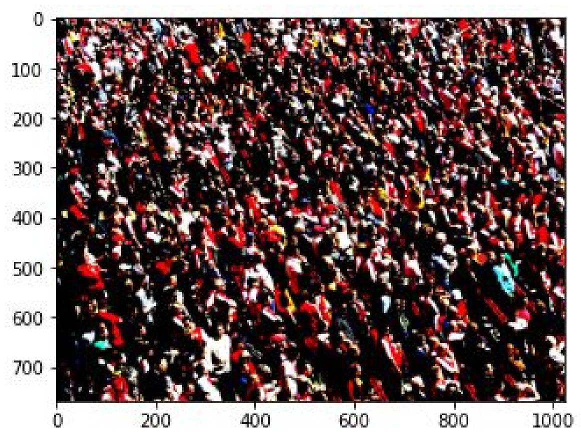
In this project, we have worked with a single dataset called ShangaiTech [1], which consists of two parts, part A and B. Part A of the ShangaiTech dataset contains 300 images for training, 182 images for testing. These images are taken from random areas on the Internet. In part B, there exists 400 images for training, and 316 for testing.

The dataset contains the actual images with different resolutions in JPG format and their ground truth values through Matlab .mat files that contain the coordinates of the people in the image.

Most of the time, we have worked with part A due to limited computational power and limited time we had to train our models. However, at the end of the project, we have trained our models with both part A and part B, and observed the results on the whole dataset.

3.1.1. Preprocessing

We have processed the data before the training by gaussian filter transformation to get the density maps of each image using the coordinates at the ground truth Matlab .mat files. We scaled the density map to easily get the total number of people by summing up the whole map. We have used these density maps for the training as ground truth values. While creating density maps, annotations on the heads of people were used.



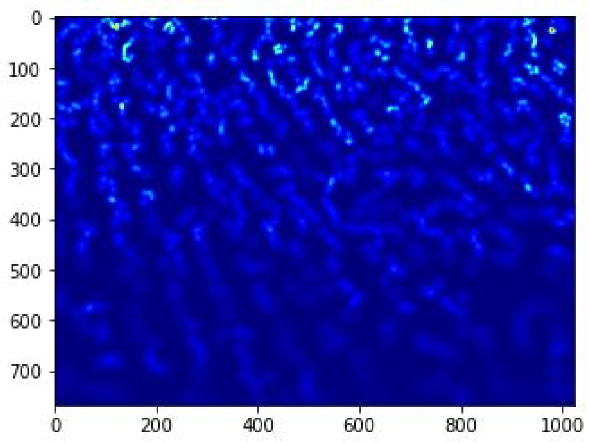
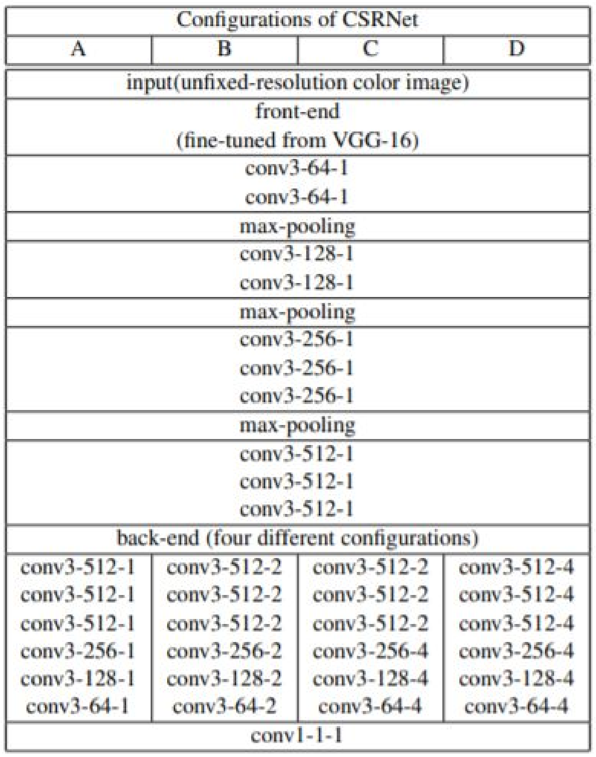


Figure 1: The Original Image and Its Density Map

3.2. CSRNet



Configurations of CSRNet			
A	B	C	D
input(unfixed-resolution color image)			
front-end (fine-tuned from VGG-16)			
conv3-64-1			
conv3-64-1			
max-pooling			
conv3-128-1			
conv3-128-1			
max-pooling			
conv3-256-1			
conv3-256-1			
conv3-256-1			
max-pooling			
conv3-512-1			
conv3-512-1			
conv3-512-1			
back-end (four different configurations)			
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-256-1	conv3-256-2	conv3-256-4	conv3-256-4
conv3-128-1	conv3-128-2	conv3-128-4	conv3-128-4
conv3-64-1	conv3-64-2	conv3-64-4	conv3-64-4
conv1-1-1			

Figure 2: The Structure of CSRNet [2]

The critical difference of this model from other CNN based crowd counting methods is to use dilated convolutional layers to increase the accuracy of the model. It has VGG-16 layers in the front-end part and additional dilated convolutional layers at the back end part as can be seen in Figure 2. Dilated convolution can be defined as the equation (1).

$$y(m,n) = \sum_{i=1}^{M} \sum_{j=1}^{N} x(m + r \times i, n + r \times j) w(i,j)$$

(1) [2]

It is important to notice that the equality becomes the same as normal convolution when we set r as 1. Dilated convolution is needed, because the traditional pooling layer reduces spatial resolution while deconvolutional layers may cause an increase in the complexity [2]. Dilated convolution leads to alternating pooling and convolutional layers.

While working with this model, we have used Adam optimizer and binary cross entropy as our loss function.

3.2.1. Dilation Rate

Increase in dilation rate reduces the spatial information that is learned from the train data by changing the convolution kernel. As can be seen in Figure 3, the adjacent values don't attend to the convolution when the dilation rate is greater than 1. When we increase the dilation rate, the elements of the kernel are getting apart from each other. Therefore, the spatial information reduces and the overfitting to the data is prevented.

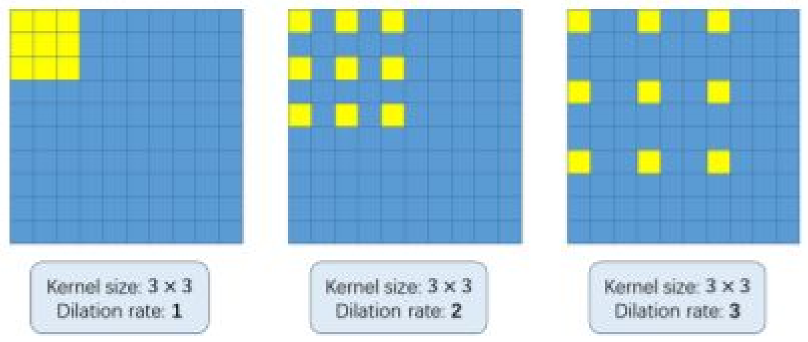


Figure 4: Different Kernels with Size 3x3 and Dilation Rates 1, 2, and 3

3.3. MCNN

In order to create an adaptive model to people with different head sizes, filters of different sizes are used to model the density maps, because it is not possible to identify the scale of the heads with filters of the same size [3]. Figure 4 shows the general structure of the MCNN model.

While working with this model, we have used MSE as our loss function.

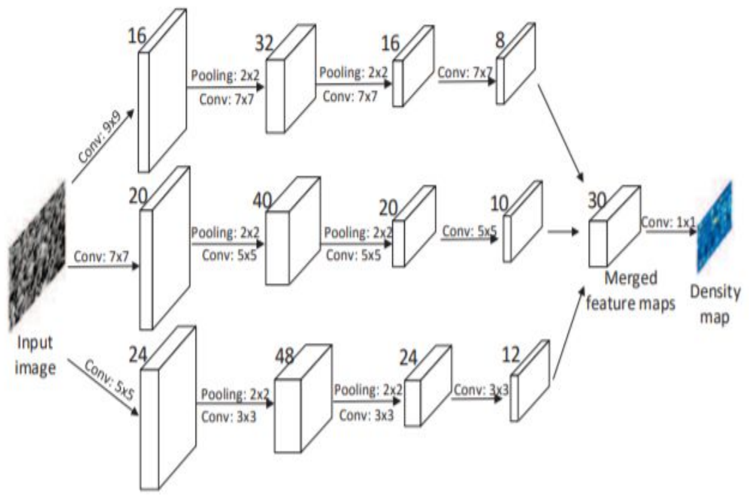


Figure 5: The Structure of MCNN [3]

3.4. Data Augmentation:

We have used the CCaugmentation [4] framework in Python that is designed for the ShanghaiTech dataset. It supports some transformations such as crop, grayscale, flip left right, etc for the images in the ShanghaiTech dataset. By this framework, we have increased our dataset without finding and labeling new images.

4. Results

Initially, we were evaluating our models with MAE, and MSE. However, we have thought that it is not realistic to measure the performance of the model using these parameters, because when the image contains approximately 1500 people, the deviation of 100 people does not seem to be very important. However, if the image contains 200 people, then, this becomes a large error rate. Therefore, we have decided to evaluate our models based on mean absolute percentage error (MAPE) and modified mean absolute error (MMAE) by the following equation.

Modified mean absolute error (MMAE): $\frac{\sum_{i=0}^{n} |A_t - P_t|}{\sum_{i=0}^{n} |A_t|}$ where A_t and P_t are ground truth and predicted value for the number of people in an image, respectively.

An increase in the MMAE means that the average error for a person in any image is increased. Moreover, an increase in the MAPE means that the average error for an image is increased. Furthermore, if MMAE is better than the MAPE, this concludes that the model is better on more crowded images.

In our project, we have tried to find answers to the following questions:

- Does making density map estimations using Convolutional Neural Network give good results while counting dense crowds?
- There are two well-known models in this area, CSRNet, and MCNN. Which model performs better results? Which model works faster? Why?
- CSRNet uses dilational convolutional neural networks. How does changing dilation rate affect the model performance?
- When it comes to convolutional neural networks, it is very important to work with large datasets in order to get good results. Therefore, applying data augmentation methods is important to be able to efficiently use existing data. However, does data augmentation really improve the model accuracy?
- How does using larger datasets affect the performance? Is it worth it to use additional computational power and time on training with larger dataset?

4.1. CSRNet

- First of all, we have trained our CSRNet model with a relatively smaller dataset, which consists of the images from Part A in the dataset. We obtained the following results for dilation rate 1:

  MAPE : 29.48%

  MMAE : 23.14%

- Then, we have trained the CSRNet model with dilation rate 1 using the train data of part A, part B, and the test data of part B. Then we have tested the model with the test data of part A. In other words, we have trained the model with 1016 and tested with 182 different original images. We have obtained a Training Loss per Epoch graph as shown by Figure x. To compare with previous experiment, we got following results:

  MAPE as 25.44%

MMAE as 20.84%.

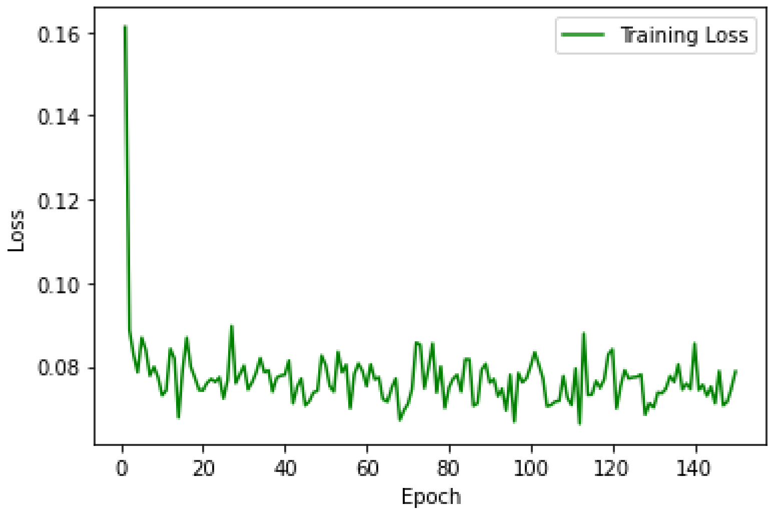


Figure 6: The Training Loss of Each Epoch While Training CSRNet with 1016 Images

- We have obtained a 4% decrease in MAPE. This is actually a very large difference in terms of model performance. However, training time of the data was 2.5 times larger than the previous one, which means that it takes 9-10 hours to train.

4.1.1. Manipulating the Dilation Rate in CSRNet

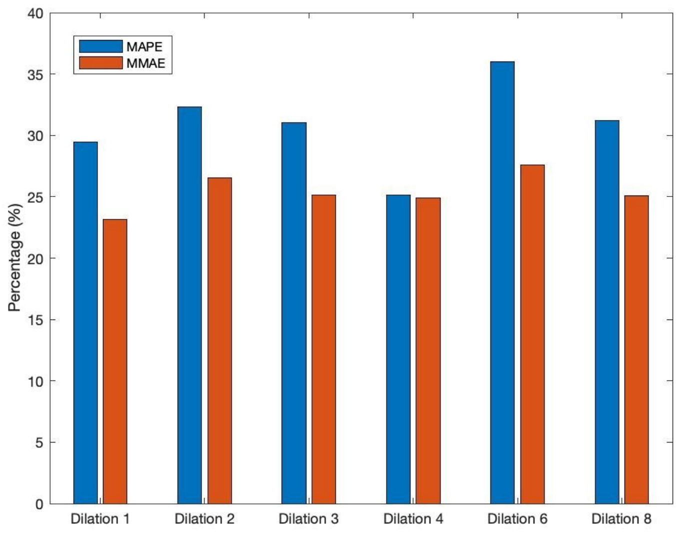


Figure 7: The MAPE and MMAE Errors for Different Dilation Values

- We have trained the CSRNet model with different dilation rates. As can be seen from Figure 2, we get the best result when the dilation rate is 1. According to the paper of the CSRNet [2], the best dilation value in set {1, 2, 3, 4} is 2. They have got the minimum error value by training with 2 as dilation value. However, based on our work, the best dilation value in set {1, 2, 3, 4, 6, 8} is 1. We have got the best error values for both MAPE and MMAE by training with 1 as dilation value.

4.1.2. Data Augmentation

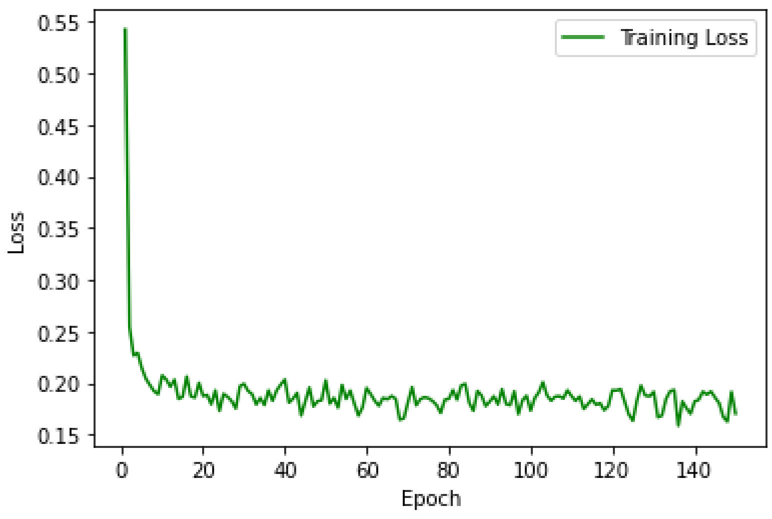


Figure 8: The MSE of Each Epoch While Training CSRNet with Augmented Part A Images by Cropping

We have augmented part A train data by cropping existing images randomly using CCaugmentation [4] library in Python. However, the errors increased after the data augmentation. As can be seen from Figure 8, the MSE never drops below 0.15 which is greater than the MSE of the training without augmentation. Before the augmentation, we got MAPE as 29.48% and MMAE as 23.14%. After the augmentation by cropping images, we got MAPE as 32.80% and MMAE as 25.14%. Therefore, it can be concluded that the augmentation by just cropping images didn't work for our model.

We have also augmented part A train data by just flipping the images left to right. However, after the training with augmented data, we got MAPE as 30.74% and MMAE as 25.14% which are also greater than the training without augmentation.

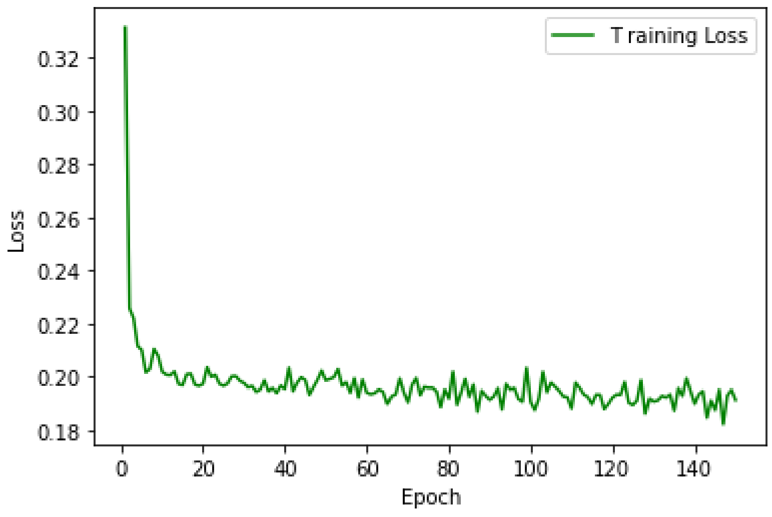


Figure 9: The MSE of Each Epoch While Training CSRNet with Augmented Data by Cropping and Making The Part A Images Grayscale

Then we have tried to augment part A train data by making the existing images grayscale and we have obtained a training loss graph in Figure x. Before the augmentation, we had MAPE as 29.48% and MMAE as 23.14%. However, with the augmentation we got MAPE as 31.56% and MMAE as 23.85%. It was actually very interesting to see that data augmentation has increased the error rate for CSRNet.

MAPE :	31.56%
APE :	23.85%
MSE:	22678.25
MAE :	102.83

Therefore, the augmentation didn't produce any improvements to our CSRNet model.

4.2. MCNN

- First of all, we have trained MCNN model with images from Part A and obtained the following training loss per epoch graph. It was interesting to see that we have obtained a graph with sharp ascents and descents this time.

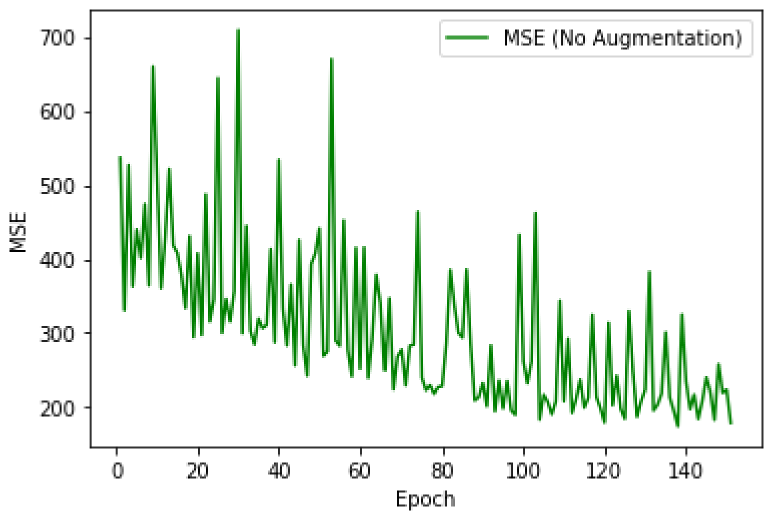


Figure 10: The MSE of Each Epoch While Training MCNN with Part A Images

- Then, we have tested the model with the test data of part A. In other words, we have trained the model with 300 and tested with 182 different images. As can be seen in As a result, we got MAPE as 73.69%, MMAE as 49.74%, MAE as 215.88, and MSE as 316.00. Actually, this model has given us very large errors compared to the CSRNet Model whose errors were MAPE : 29.48% and MMAE : 23.14%. However, the training time of this model was very short compared to the CSRNet model. It can be considered as a trade-off between accuracy and time to train.
- We have trained our model with a larger dataset which contains 1016 images to train and 182 images to test. We have obtained the following training loss per epoch graph and following results:

MAE: 180.00

MSE: 264.90

MAPE: 53.03%

MMAE: 41.47%

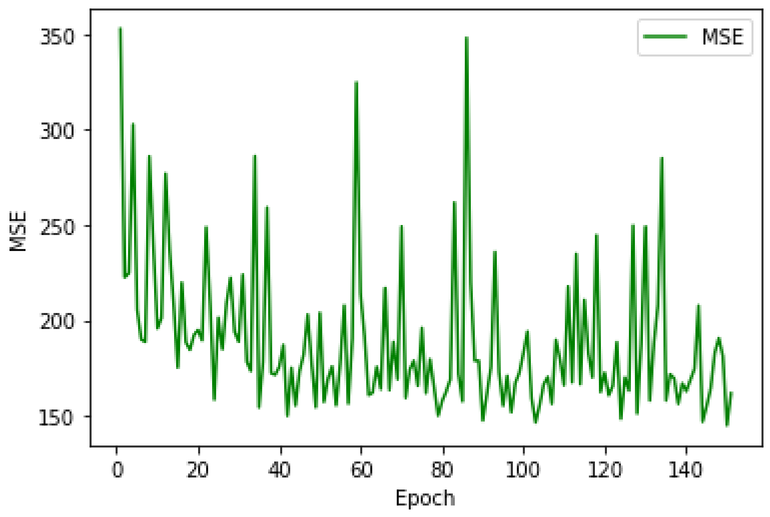


Figure 10: The MSE of Each Epoch While Training with Large Dataset with 1086 images

- Training with a larger dataset affected the MCNN model's performance very much. We have observed a decrease in MAPE approximately 20%. This was a very large difference compared to CSRNet's 4%.
- Then, we have augmented part A train data by cropping and making the existing images grayscale and randomly cropping. We get more than 2700 images by this augmentation for the training. Before the augmentation, we got MAPE as 73.69%, MMAE as 49.74%, MAE as 215.88 and MSE as 316.00. After the augmentation, we got MAPE as 33.76% and MMAE as 31.00%. Moreover, we got MAE as 134.56 and MSE as 211.21. Therefore, it can be concluded that the augmentation by cropping and making the images grayscale for training the model clearly decreased the errors and improved our MCNN model.

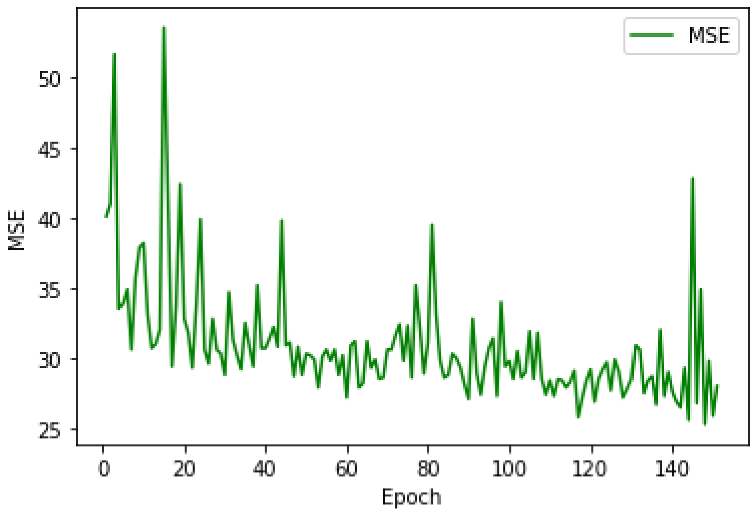


Figure 11: The MSE of Each Epoch While Training MCNN with Augmented Data by Cropping and Making The Part A Images Grayscale

5. Discussion

5.1. Comparison of the CSRNet and MCNN

Both models use density maps obtained from images. Compared to CSRNet, MCNN takes less time to train however, its error rate is larger, because CSRNet contains deeper convolutional layers and it is a more advanced model.

5.2. Data Augmentation Results

Using augmented data to train our MCNN model improved the model tremendously. However, we couldn't observe the same effects when training our CSRNet model with the augmented data. It naturally prolonged the training time but without giving any meaningful performance gain.

5.3. Small Dataset vs Large Dataset

As we expected, using larger datasets while training both models ended in improved accuracy and performance of the models. It increases the training time naturally, however for CSRNet improvements we have observed were much more effective than improvements obtained in the MCNN model.

6. Conclusion

In conclusion, CSRNet performed much better than MCNN but it also took a considerably longer amount of time to train. Changing the dilation rate while training the CSRNet didn’t prove as useful and we spent a lot of time trying to figure the effect of the dilation rate.

If the models could be adjusted for live footage as opposed to static images, it could be utilized for Covid-19 related applications such as detecting whether a certain area has reached the number of people threshold etc.

7. Appendix

Dilational rate and CSRNet experiments → Emre, Berk

Data Augmentation, MCNN experiments → Aybüke, Yağmur

References

[1] Thien, T. (2019, June 13). ShanghaiTech. Retrieved December 20, 2020, from https://www.kaggle.com/tthien/shanghaitech

[2] Li, Y., Zhang, X., & Chen, D. (2018). CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. doi:10.1109/cvpr.2018.00120

[3] Zhang, Y., Zhou, D., Chen, S., Gao, S., & Ma, Y. (2016). Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.70

[4] CCaugmentation. Retrieved December 20, 2020, from https://pypi.org/project/ccaugmentation/