Counting People in Crowds Using Multiple Column Neural Networks

Christian Massao Konishi and Helio Pedrini
Institute of Computing, University of Campinas, Campinas, Brazil

Keywords: Crowd Counting, Generative Adversarial Networks, Deep Learning, Activation Maps.

Abstract: Crowd counting through images is a research field of great interest for its various applications, such as surveillance camera images monitoring, urban planning. In this work, a model (MCNN-U) based on Generative Adversarial Networks (GANs) with Wasserstein cost and Multiple Column Neural Networks (MCNNs) is proposed to obtain better estimates of the number of people. The model was evaluated using two crowd counting databases, UCF-CC-50 and ShanghaiTech. In the first database, the reduction in the mean absolute error was greater than 30%, whereas the gains in efficiency were smaller in the second database. An adaptation of the LayerCAM method was also proposed for the crowd counter network visualization.

1 INTRODUCTION

Obtaining an adequate estimate of the number of people present in an image has several practical applications. Counting a few tens of individuals is simple enough to be done manually, however, in large crowds, such as public manifestations, musical events and sporting events, using a crowd counting model may not rarely be the only viable option, allowing for better urban planning, event planning, and surveillance of crowds.

An intuitive way to model an object counter is to train a detector and, using it, determine the amount present in the image (Li et al., 2008). However, these models cannot adequately handle large densities of people (Gao et al., 2020), because they rely on recognizing some body part, such as head or shoulders, that may be partially occluded in a crowd. Other models (Zhang et al., 2016; Lempitsky and Zisserman, 2010) do not seek to detect and localize the position of each person, they aim to calculate the quantity of objects in an image by estimating the density in a given region of the image.

One difficulty in these models is dealing with variations in image conditions, such as lighting, density, and size of the people. The use of a convolutional neural network with filters of different scales, such as a Multi Column Neural Network (MCNN) (Zhang et al., 2016) is an alternative for these scenarios, since it can handle variations in the size of people in a single image and variations caused by different image dimensions. On the other hand, a limitation of the MCNN is that its output is a density map of smaller height and width than the original image, which causes information loss, inherent to the model itself.

In this work, modifications to the MCNN were proposed, both in terms of architecture and training, aiming to obtain density maps that are more faithful to the reference maps (ground truth). For this, in addition to the neural network that estimates the density of people in the image, a second network was added, whose role is to evaluate the output of the first one when compared to real densities. This approach is an application of Generative Adversarial Networks (GANs), more precisely, the Wasserstein-GAN (Arjovsky et al., 2017), in the context of counting people in crowds by density maps. The proposed model for the estimator is based on an MCNN, but introduces a series of modifications to improve the quality of the output (Section 4), recovering the original image dimension and adding more possible connections between the various levels of the network.

2 RELATED CONCEPTS

The crowd counting problem consists in estimating the number of people present in an image or a video. Although other approaches exist (object detection, regression), the most modern models have been based on Fully Convolutional Networks (FCN) (Gao et al., 2020), a class of Convolutional Neural Networks (CNN) that does not feature densely connected layers.

Konishi, C. and Pedrini, H.
Counting People in Crowds Using Multiple Column Neural Networks.
DOI: 10.5220/0011704000003417
In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2023) - Volume 4: VISAPP*, pages 363-370
ISBN: 978-989-758-634-7; ISSN: 2184-4321

2.1 Density Maps

A Full Convolutional Network is able to estimate the number of people in an image of a crowd by producing a density map whose sum of all elements results in the appropriate count (Figure 6). To train a network capable of generating such maps, it is necessary to produce *ground truth* values for the training images.

Given an image I, of dimensions $M \times N$, with k people, the position of each individual is approximated to a single point, such that the positions of the people are $P = \{(x_1,y_2),(x_2,y_2),(x_3,y_3),\ldots,(x_k,y_k)\}$. Based on this, a map H is defined with the positions of the people.

$$H(x,y) = \begin{cases} 1, \text{ if } (x,y) \in P \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

The density map is obtained by convolution operations, applying a Gaussian filter on H. The size of the filter, as well as the value of σ, must be defined and may vary along the image or not.

2.2 Activation Maps

Visualizing and understanding the behavior of a neural network is not a trivial task. When the problem domain is images, it is possible to use the activation map of some intermediate convolutional layer of the network to observe the most important points for the model decision, since these maps preserve the spatial information.

In the image classification field, Grad-CAM (Selvaraju et al., 2017) is an algorithm capable of combining the activation maps and the gradients with respect to a network output class, using Equation 2.

$$M^c = \text{ReLU}\left(\sum_k w_k^c \cdot A_k\right) \tag{2}$$

where w_k^c is the average of the gradients with respect to class c in channel k and A_k is the activation map of a certain layer for channel k.

This approach, by weighting the activation maps by aggregating the gradient per channel, tends to lose the spatial information of the gradients. In deep layers in a classification network, activation maps usually have reduced height and width, which mitigates this problem. But in shallower layers or in networks that do not have such reduced activation maps, the LayerCAM (Jiang et al., 2021) algorithm is more appropriate (Equation 3).

$$M_{ij}^c = \text{ReLU}\left(\sum_k \text{ReLU}(g_{ij}^{kc}) \cdot A_{ij}^{kc}\right) \tag{3}$$

where g_{ij}^{kc} is the gradient at position ij of channel k of the activation map for class c. Note that this equation preserves the spatial information of the gradient and will be more appropriate for dealing with the MCNN. For this, instead of calculating the gradient for a class c, considering that the present problem is crowd counting, it is more appropriate to use the gradient of the summation of the output density map of the neural network.

3 RELATED WORK

Recent crowd counting approaches are heavily based on techniques to estimate the count of people in images through the density maps, Lempitsky and Zisserman (Lempitsky and Zisserman, 2010) were the first to apply the method. When compared to previous approaches, the proposed density maps stand out, since the other possibilities were (i) counting by object detection, which does not work well in partial occlusion and high density situations; (ii) counting by regression, which maps the images to a real number, the output of the model being the amount of people itself, this approach fails to utilize the spatial information, since the output of the network and the ground truth value are one-dimensional. In this sense, the density map solves both problems and therefore became the most employed technique.

The multiple column architecture for crowd counting was proposed by Zhang et al. (Zhang et al., 2016), in order to handle variations in the scale of the people present in the image. The work originally featured 3 columns, but this value can be modified. Quispe et al. (Quispe et al., 2020), for example, studied the use of several multi-column neural networks – varying the number of columns from 1 to 4 – and different Gaussian filters of fixed and variable sizes, proposing their own method to define the Gaussian filter used to generate the ground truth values for training.

4 METHODOLOGY

In this section, the methods employed in the experiments performed in this work are presented, containing information about the databases, algorithms, and architectures employed in the tests.

4.1 UCF-CC-50

One of the crowd counting databases used in this work is the UCF-CC-50 (Idrees et al., 2013). The database consists of 50 images of crowds of varying densities, with extremely dense regions (Figure 1) and annotations for each person's position.

Figure 1: Example of an image from the UCF-CC-50. It is possible to observe how the image presents people density variation, with regions of extreme concentration.

Due to the scarcity in the amount of images, the 5-fold cross-validation strategy was employed, dividing the original set into 5 groups, taking 4 groups for training and 1 for testing, repeating the procedure for each possible training and testing split. The results presented comprise the average of the efficiencies in each of the 5 folds, as well as the standard deviation.

4.1.1 Data Augmentation

Data augmentation processes were employed to expand the amount of data available for training in each fold. The strategies aimed to reproduce the results obtained by Quispe et al. (Quispe et al., 2020) and were retained for later testing, allowing direct comparison of models. Three forms of data augmentation were adopted: (i) using a sliding window of 256×256 pixels, which scrolls through the images with a step of 70 pixels, cropping the new figures; (ii) adding Gaussian noise (zero mean and variance 0.1) in half of the images generated in the previous step and impulsive noise (with 4% probability of affecting a pixel) in the other half, doubling the amount of samples; (iii) changes in the illumination conditions (Equation 4) of the images available after (ii), again doubling the amount of images. The original images themselves were not used.

$$f' = \begin{cases} f_i + 10, & \text{if } i \text{ é even} \\ 1.25\, f_i - 50, & \text{otherwise} \end{cases} \tag{4}$$

where f' is the output image and f_i is the i-th image available after step (ii).

4.2 ShanghaiTech

The ShanghaiTech database (Zhang et al., 2016) has 1198 images, split into two parts. The first, Part A contains 482 images obtained through the Internet, separated into training and testing, while Part B is composed of images obtained from streets of Shanghai (Figure 2).

(a) Part A (b) Part B

Figure 2: Examples of images extracted from the two parts of the ShanghaiTech database.

4.2.1 Data Augmentation

The procedures used to prepare the ShanghaiTech images were different from those adopted for the UCF-CC-50, in general, the process sought inspirations from the original work of Zhang et al. (Zhang et al., 2016), but adaptations were necessary to deal with some constraints imposed by the proposed model (Subsection 4.3). In short, the images must have the dimensions of 256×256 pixels during the training process, a condition that allows the mini-batch processing and the use of the critical network.

The general idea of this data augmentation process consists of making 5 cutouts of the original image, one for each corner of the figure and a central one, each cutout with the dimensions of 512×512. In addition, for each one of the 5 new images, a horizontal flip was made, totaling 10 copies. Then, each of the images was resized to 256×256 pixels, reaching then the desired dimension for training.

There is a rare but actually occurring case at the database, when an image has a height or width less than 512 pixels, in which case the cropping process would fail. To handle this case, images with height or width less than 614 pixels – about 120% of 512 pixels – were resized to have at least this amount of pixels in their dimensions, but keeping the same proportion of height and width, avoiding distortions. Once this treatment was done, the procedure as described above was applied normally. Some examples of the results can be visualized in Figure 3.

(a) Part A (b) Part B

Figure 3: Examples of images extracted from the two parts of the ShanghaiTech database after the data augmentation process.

For the test images, it is not necessary to have the dimensions of 256×256 pixels, so it was just resized, halving the amount of pixels in height and width, in a similar way to training.

4.3 Neural Network Architectures

The model used has two deep neural networks – just as in a standard GAN model –, these being the critical network (Figure 4) and a multi-channel neural network, which will be referred to as MCNN-U (Figure 5), which is analogous to the generative network of a Wasserstein-GAN, but whose input is a monochromatic image rather than noise, as well as using transposed convolutions to preserve the dimension of the density map and skip connections to increase the complexity of the model, differentiating it from previous (Quispe et al., 2020) models.

The MCNN employed is composed of 4 columns – referred to as U-columns – of different filter sizes (Figure 5). Each column consists of convolution and max pooling operations that reduce the size of the activation map, followed by transposed convolution operations that recover the original map size (upscale). In addition, skip connections have been added connecting the convolutional layers and the upscale layers, allowing the architecture to combine activations from different depths of the network. It is worth mentioning that the 1×1 convolutions used in the skip connections were employed with the purpose of reducing the amount of transmitted channels, which would make the model heavier, moreover, without this reduction there would be more data from shallow layers than deep layers.

4.4 Density Maps

The task of the MCNN-U is to produce a density map whose summation corresponds to the amount of people in the image. To produce the ground truth values, given an image of dimensions $M\times N$, a null matrix of the same size is created. At the position of each person, the value 1 is placed in the matrix, and finally, a convolution is performed with a Gaussian filter (Figure 6), with dimensions 15×15, $\sigma = 15$.

4.5 Cost Function

The cost function used for training the described neural networks can be divided into two parts, one that corresponds to the basic goal of decreasing the distance between the output of the MCNN-U – the MCNN-U network will be denoted G, hence the output of G for an image I is denoted $G(I)$ – and the ground truth value (gt); and another part that concerns the Wasserstein cost of the adversarial network model. The critical network will be denoted as C.

The distance between the density distribution maps is given by the root mean square error of the matrix values:

$$L_{\mathrm{MSE}} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}\left[G(I_i)(x,y) - gt(x,y)\right]^2 \tag{5}$$

where M is the width and N is the height of the density map, while m is the size of the batch.

On the other hand, the Wasserstein cost for the generative network is given by:

$$L_W = -\frac{1}{m}\sum_{i=1}^{m} C(G(I_i)) \tag{6}$$

Combining the two costs gives the cost of the generative network:

$$L_G = L_{\mathrm{MSE}} + \alpha L_W \tag{7}$$

α being a hyperparameter to be decided.

The cost for the critical network is given by:

$$L_C = \frac{1}{m}\sum_{i=1}^{m} C(gt_i) - \frac{1}{m}\sum_{i=1}^{m} C(G(I_i)) \tag{8}$$

The method goal is to minimize the value of L_G and maximize that of L_C.

4.6 Test Configuration

This subsection presents how the training and model evaluation were conducted. Different versions of the model with variations in hyperparameters were evaluated, but only the final version will be presented in this summary.

The Wasserstein cost was applied through the value of $\alpha = 3{,}500$ (Equation 7). In addition, the density maps were multiplied by 16,000. For each one of the UCF-CC-50 training sets, 1,000 epochs were run, with a batch size of 32 images. For ShaghaiTech, Parts A and B, the value of 1,500 epochs was employed, with a batch size of 64 images.

The Rectified Adam (Liu et al., 2020) optimizer was adopted, with learning rate $lr = 1\cdot10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The value of *ncritic*, that is, the number of times the critical network receives data and is optimized for each pass through of the generative network, was set to 3.

The algorithms were all executed using virtual machines from Google Colaboratory. The machines typically provide one core of an Intel Xeon (variable model), about 12 GB of RAM and a graphics processing unit (GPU) that can vary between an Nvidia Tesla K80, an Nvidia Tesla P100 or an Nvidia Tesla T4.

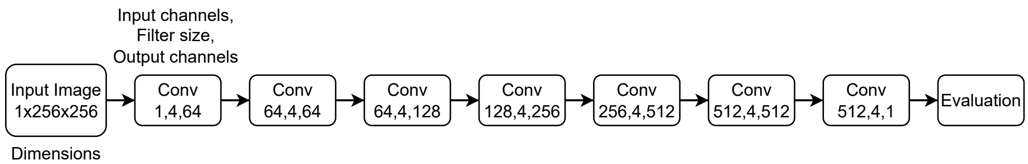


Figure 4: Critical network architecture. The activation function used is the leaky ReLU ($\alpha = 0.02$) and the convolutions have parameters 4, 2 and 1 for the size of the kernel, stride and padding, respectively, the batch normalization technique is also employed. The output layer has no activation function and the convolution has parameters 4, 1 and 0.

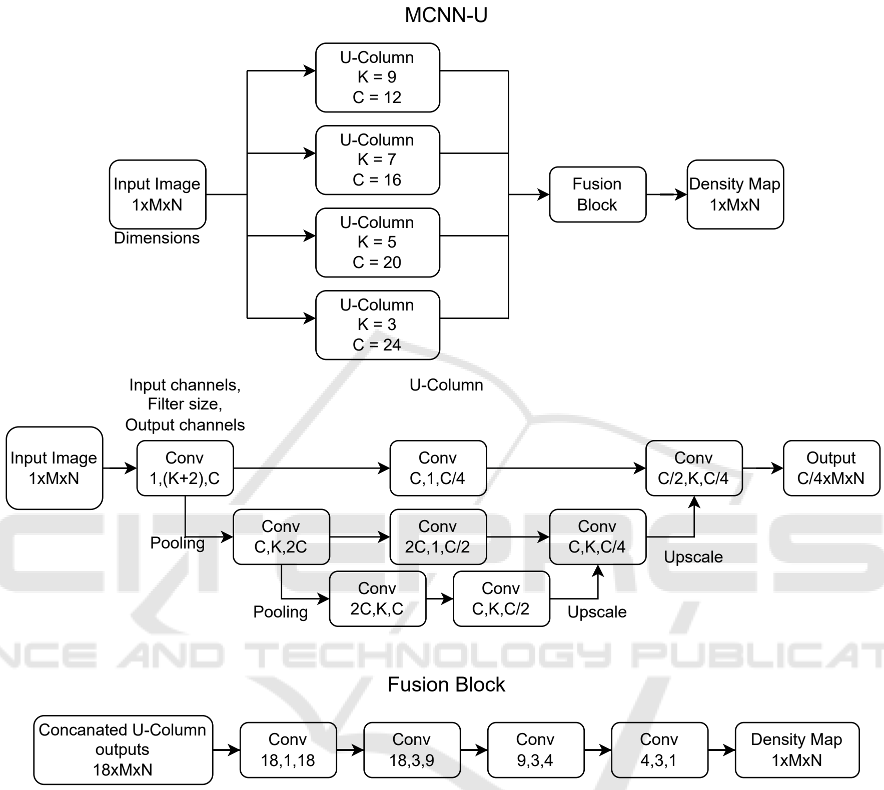


Figure 5: MCNN-U Description. The tensor dimension is represented in the format <channels>$\times$<width>$\times$<height>, while the convolutions, in the format <input channels>, <kernel size>, <output channels>, using an appropriate padding to maintain the dimensions of the activation map; each convolution is followed by a ReLU activation function, except for the output layer. The max pooling operation is used to halve the height and width dimensions, while upscale recovers the original dimension with transposed convolutions with a kernel size $= 2$, stride $= 2$ and zero padding. 1×1 convolutions were used in the skip connections to decrease the amount of transmitted channels.

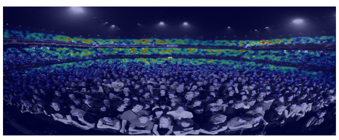


Figure 6: Visualization of the density distribution map as a heat map, overlaid with its original image, for the UCF-CC-50 database.

4.7 Performance Metrics

To measure the performance (effectiveness) of the tested networks, the sum of the density map of the output of the generative network is compared with the map generated through the Gaussian filter. To quantify the difference between the two counts, two measures were employed:

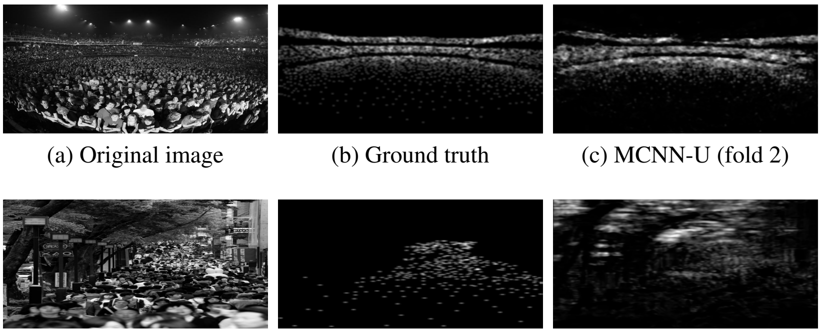
- Mean absolute error:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} \left|cnt_i - cnt_i'\right| \tag{9}$$

- Root mean square error:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (cnt_i - cnt_i')^2} \tag{10}$$

where N is the number of images in the test set, cnt_i is the correct count of people in image i, and cnt_i' is the



(a) Original image (b) Ground truth (c) MCNN-U (fold 2)

(d) Original image (e) Ground truth (f) MCNN-U fold 3)

Figure 7: Visualization of the density maps produced using the ground truth values and the MCNN-U for the UCF-CC-50 database.

Table 1: Efficiencies obtained by the MCNN-U model.

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average	Standard Deviation
MAE	424.5566	204.7662	197.8882	229.6904	225.2515	256.4306	84.9117
RMSE	709.0030	294.2144	323.6123	321.3106	345.6961	398.7673	155.9756

count made from the neural network output. Note that these values only evaluate the count result and not the content of the density distribution maps themselves.

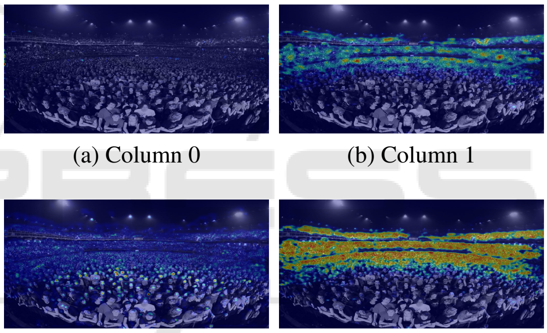
For the UCF-CC-50 case in particular, the two metrics are calculated for each of the 5 folds, and the final result is given by their average. The standard deviation was also calculated, as it represents how homogeneous the model's effectiveness was for each fold.

5 EXPERIMENTAL RESULTS

The ground truth density maps and those produced by the MCNN-U for the same image can be viewed by placing them side by side for comparison purposes (Figure 7). In the upper figures, the network result seems to match what was expected, but in the background, some regions of high density of people were not identified – which can also be confirmed in Figure 8. In the lower images, a more critical case is presented, with a large number of false positives in the tree leaf region.

Table 2: Comparison of the effectiveness achieved by the MCNN-U in the UCF-CC-50 database in relation to other models in the literature.

Model	Mean MAE	Mean RMSE
MCNN (Zhang et al., 2016)	377.6	509.1
MSNN_3 (Quispe et al., 2020)	374.0	554.6
CP-CNN (Sindagi and Patel, 2017)	295.8	320.9
MCNN-U	256.4	398.8
CAN (Liu et al., 2019)	212.2	243.7



(a) Column 0 (b) Column 1

(c) Column 2 (d) Column 3

Figure 8: Visualization of the activation maps for each of the columns of the MCNN-U in the UCF-CC-50 database, column 0 (a) being the one with the largest filter size, and column 3 (d) being the smallest. A Gaussian filter was applied to the LayerCAM output to improve the map visualization.

Using the LayerCAM described previously, it is possible to independently observe the behavior of each column of the MCNN-U (Figure 8). It is noticeable that each column was more (or less) activated in different regions of the image, due to the variation in density of people, column 0 was not activated, while column 1 and especially column 3 focused on the region of higher density, while column 2 identified the larger people, with emphasis on the region of the heads. The results obtained after training the MCNN-U on the UCF-CC-50 database, for the hyperparameters defined in Subsection 4.6 were compiled in Table 1, separated by fold.

The effectiveness of the model was relatively constant across all but the first fold. This pattern was

Table 3: Comparison of the effectiveness achieved by the MCNN-U in the ShanghaiTech database in relation to other models in the literature.

Model	Part A		Part B	
	Mean MAE	Mean RMSE	Mean MAE	Mean RMSE
$MSNN_4$ (Quispe et al., 2020)	163.4	242.7	34.5	57.7
MCNN (Zhang et al., 2016)	110.2	173.2	26.4	41.3
MCNN-U	105.5	152.4	18.3	30.0
CP-CNN (Sindagi and Patel, 2017)	73.6	106.4	20.1	30.1
CAN (Liu et al., 2019)	62.3	100.0	7.8	12.2

repeated throughout previous model tests, this being the most challenging partition of the database, possibly due to the high density of the test images. In order to be able to evaluate the effectiveness achieved by MCNN-U, the performance metrics were compared with other results from the literature (Table 2).

The efficiency obtained represents a leap when compared to the original MCNN, demonstrating that the modifications applied were indeed appropriate to the model. In relation to the approaches of the literature, the results were competitive, but more complex approaches that seek to evaluate the image context obtained higher efficiencies, which may, on the other hand, result in more complex training.

For the ShanghaiTech database, on the other hand, Part A has images with people densities comparable to the UCF-CC-50 database, while Part B, despite having photos of busy streets, the people counts are generally lower, with considerable portions of the images without people. It is especially interesting to observe the behavior of the MCNN-U for these cases of less dense images (Figure 9).

(a) Column 0

(b) Column 1

(c) Column 2

(d) Column 3

Figure 9: Visualization of the activation maps for each one of the MCNN-U columns in the ShanghaiTech Part B database, column 0 (a) being the largest filter size, and column 3 (d) being the smallest. A Gaussian filter was applied to the LayerCAM output to improve the visualization of the map.

It can be seen in this case that, unlike the activations for UCF-CC-50, here all columns were activated for some part of the image, while in Figure 8, the column with the largest filter was not activated. One concern that existed was that many false positives would occur in the empty regions of the image, but looking at this result and others, this does not seem to be the case. Still on Part B, training on this database was considerably more unstable than Part A, or any other fold of the UCF-CC-50, with the cost function showing sudden spikes during training (Table 3). Whether this was caused by a numerical instability of the RAdam optimizer or whether it is an intrinsic characteristic of the model combined with the database, defining a space with many local minima, is still an uncertain aspect, further tests with other configurations of the optimizer are needed.

Unlike what was observed with the UCF-CC-50 tests, the increase in efficiency obtained with the MCNN-U when compared to the MCNN model was considerably lower, especially in Part A. A possible reason for this difference lies in the data augmentation process. In fact, the model of Zhang et al. was trained under different conditions, the data treatment was not the same, as well as the training process. The main difference would be in the fact that the MCNN was trained one image at a time, without mini-batch, and this allows the training images to have different dimensions, for example. Other factors such as optimizer settings may also affect, and the fact that Part B training was more unstable, as mentioned earlier, may point in the direction that the data augmentation process or the ShanghaiTech training guidelines themselves need more adjustment for MCNN-U.

6 CONCLUSIONS

The effectiveness obtained by the MCNN-U was considerably higher compared to the original MCNN in the best case scenario. The proposed changes to the model were tested incrementally and the experiments with ShanghaiTech Part B indicated that there is room for improvement in the model in terms of stability, possibly with further testing on other optimizers or by adding modifications to the network to achieve smoother convergence.

REFERENCES

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein Generative Adversarial Networks. In Precup, D. and Teh, Y. W., editors, *34th International Conference on Machine Learning*, volume 70, pages 214–223. PMLR.

Gao, G., Gao, J., Liu, Q., Wang, Q., and Wang, Y. (2020). CNN-based Density Estimation and Crowd Counting: A Survey. *arXiv preprint arXiv:2003.12783*, pages 1–25.

Idrees, H., Saleemi, I., Seibert, C., and Shah, M. (2013). Multi-source Multi-scale Counting in Extremely Dense Crowd Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2547–2554.

Jiang, P.-T., Zhang, C.-B., Hou, Q., Cheng, M.-M., and Wei, Y. (2021). LayerCAM: Exploring Hierarchical Class Activation Maps for Localization. *IEEE Transactions on Image Processing*, 30:5875–5888.

Lempitsky, V. and Zisserman, A. (2010). Learning to Count Objects in Images. *Advances in Neural Information Processing Systems*, 23:1324–1332.

Li, M., Zhang, Z., Huang, K., and Tan, T. (2008). Estimating the Number of People in Crowded Scenes by Mid based Foreground Segmentation and Head-shoulder Detection. In *19th International Conference on Pattern Recognition*, pages 1–4. IEEE.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. (2020). On the Variance of the Adaptive Learning Rate and Beyond. In *Eighth International Conference on Learning Representations*, pages 1–14.

Liu, W., Salzmann, M., and Fua, P. (2019). Context-Aware Crowd Counting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5099–5108.

Quispe, R., Ttito, D., Rivera, A., and Pedrini, H. (2020). Multi-Stream Networks and Ground Truth Generation for Crowd Counting. *International Journal of Electrical and Computer Engineering Systems*, 11(1):33–41.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. In *IEEE International Conference on Computer Vision*, pages 618–626.

Sindagi, V. A. and Patel, V. M. (2017). Generating High-Quality Crowd Density Maps Using Contextual Pyramid CNNs. In *IEEE International Conference on Computer Vision*, pages 1879–1888.

Zhang, Y., Zhou, D., Chen, S., Gao, S., and Ma, Y. (2016). Single-image Crowd Counting via Multi-column Convolutional Neural Network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 589–597.