Densely Connected Feature Pyramid Network for Crowd Counting

Ma Jiawei
jiawei.ma@studenti.unipd.it

Abstract

Crowd analysis is a broad multidisciplinary study that involves many scientific fields such as mathematics and psychology ect. This is an attractive subject also for the computer vision community that is committed to create a machine algorithm to automatically detect crowd flow from video frames. To achieve this one can design an image processing pipeline either to estimate crowd density (number of heads) or even to annotate critical areas. There are two main approaches; one is using classic computer vision techniques and other is to leverage novel deep learning architectures.
In this work we take the second direction and we would like to give our contribution by proposing a promising deep learning architecture based on the idea of feature pyramids, it is recently introduced in the object detection domain powered by deep models but already applied in some classic computer vision algorithm such as SIFT (Scale Invariant Feature Transform).

1. Introduction

Crowd counting is a computer vision task that aims to estimate the *crowd density* within a supervised area, in this work the objects in movement are people flow.
The task is very important in those cases when the maximum occupancy of a room or building must be limited due to safety concerns, for instance, in some public places the number of emergency exits installed is based on the maximum crowd density. Recently, crowd monitoring is becoming more and more a heated topic because of COVID-19, so it's useful to limit the virus spreading.
Early studies were focus on traditional computer vision techniques that required complex *feature engineering* and the results were often not satisfactory [8]. Thanks to new advances in *Deep Learning*, more powerful vision model architectures are now available, especially those based on CNNs, theses are so popular also because of hardware acceleration (GPUs) and *big data*. So, in this work we try to implement outstanding neural network architectures that allow us to predict/estimate the number of persons in a 2-D image, the images are some video frames recorded by video surveillance systems and the contexts are typically public manifestations, sport events or shopping malls etc. Even if there are not large public available datasets in this work we propose some promising models built starting from the existing deep networks implementing breakthrough ideas inspired by recent achievements in the object detection research field. The final models can be extended to other domains, such as biology for counting number of cells in a micro organism.

2. Related work

There are several approaches to tackle this problem and well explained in [8], there is also a quite exhaustive list of these techniques reported in [1]. Briefly, they are based on traditional computer vision algorithms such as SIFT (Scale Invariant Feature Transform) combined with some statistical methods (regression methods).
Here we concentrate on CNN based neural network approach. Many studies were conducted by using CNNs and they were even more ambitious; the desired results were not only a simple count estimation but also a *density map* visualizing the most crowded areas in the images, the advantage of having this further information, as naturally returned by the classic computer vision methods, is that the maps can be used to detect anomalies in the area supervised and this is related to a specific research area of anomaly detection [7].
Two works caught our attention, both are CNN based models; the first one introduced the *Multi-Column Neural Network* (MCNN) [13] and the second one presented a model architecture named *Congested Scene Recognition Network* (CSRNet) [4]. Specifically, the first model was designed by using three CNN branches (columns) to produce the density maps and then the algorithm counts the heads by integration, the authors claimed that multi-branches architecture is needed since the images usually contain heads of very different size, hence filters with receptive fields of the same size are unlikely to capture characteristics of crowd density at different scales. The more recent design CSRNet, leverages VGG-16 [10] as feature extractor, intended to improve the state-of-art approach at that time such as MCNNs by reducing the model complexity, so their resulting model

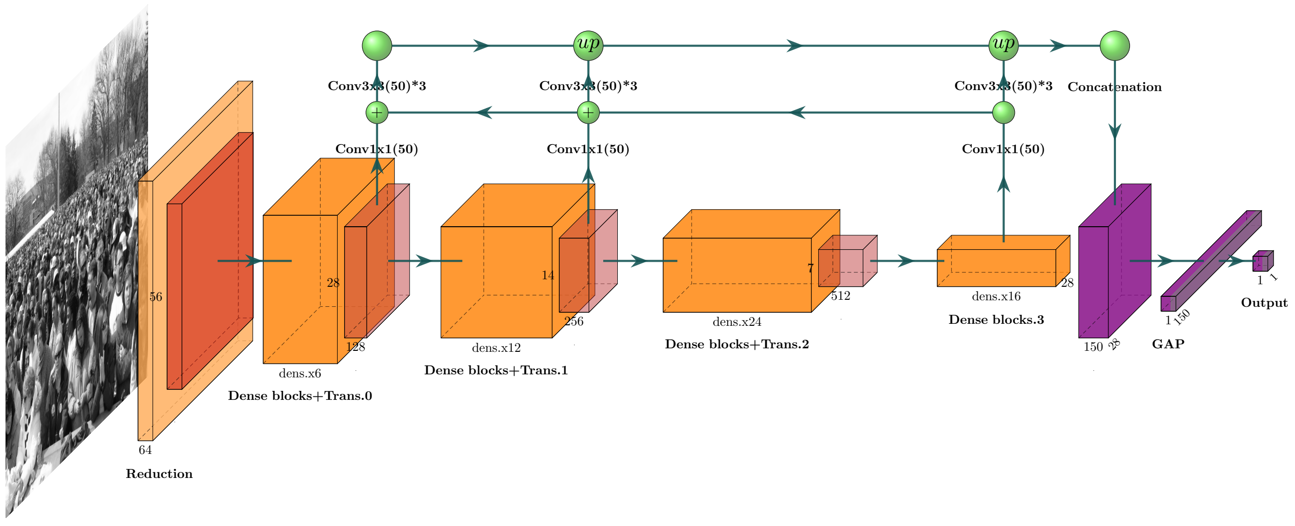


Figure 1: **FPN121** architecture overview. DenseNet121 backbone added top-down pathway and lateral connections.

should be more efficient and accurate. The key idea of the CSRNet is the *dilated convolutional layer* and it can be seen as a deconvolutional layer but the authors proved the new component had better performance showing more detailed information in the reconstructed images. An common strategy that both works above implemented was the generation of ground truth density maps by using *geometry-adaptive Gaussian kernels*, the details can be found in the MCNN paper.

Our approach to the problem is simpler; we would like to better estimate as much as possible the ground truth counting without generating any density maps. The main advantage is that we can avoid to generate the density maps and an additional step to estimate the count numbers, both these operations require a careful analysis of the problem and data, as consequence, this can affect negatively on the final models' performance. Ironically, the drawback is due to the absence of the previous procedures; as in many other deep learning researches, what we present here are *black box* models. However, our models are relatively small in terms of network parameters and we obtained results on the available datasets similar to what the other algorithms achieved, even better in some of the cases and if not we can still argue that we reduced the images' resolution in order fit everything in a single 16 GB GPU, so theoretically, there is still room for improvement.

3. Feature pyramid based approach

Our solution is inspired by the main idea of the MCNNs and by the original work on *Feature Pyramid Networks* (FPNs) [6]; the key concept of the FPNs can be applied to address the the problem of detecting objects at different scales. So, we are convinced that implementing feature pyramids inside deep networks can be very helpful to achieve good prediction results. The FPN is a very general framework, indeed, in the original paper the authors developed models to solve a variety of problems such as object detection and segmentation. It's worthy to mention the last enhancement of this innovative idea that is *bi-directional FPN* (BiFPN) [12]. In the same paper this new design was applied to the EfficientNets (backbones) [11] properly modified (scaling parameters were different) for object detection and the final models are called *EfficientDets*.

In our study we only experimented with FPN based architectures, specifically, the backbones we consider are two: *DenseNet121* and *DenseNet169*. We will show that implementing feature pyramids can really boost the starting models, either training phase acceleration or prediction precision improvement.

3.1. DenseNet

DenseNets [3] are very popular in deep learning and computer vision communities because of their good capability in feature extraction and high model efficiency. In the official paper they described this new model architecture as the extension of the *ResNets* [2]; the training is faster, more efficient in terms of network parameters and better performance. Indeed, the key idea of DenseNet is propagating *skip connections via concatenation* throughout the network and this should be helpful to impose the layers to more focus on relevant features.

The reasons why we chose DenseNets as backbones were that, firstly, they are efficient and general-purpose models,

secondly, FPN's components could be easily embedded in these structures because the output dimensions meet the size requirements, otherwise, using other architectures such as ResNets or Inception networks, we should suitably modify some of the layers inside the networks. Furthermore, the efficiency had the same importance due to scarce computational resources available.
There are three versions of DenseNet; DenseNet121, DenseNet169 and DenseNet202. The only difference between them is the number of the filters in each dense block (there are totally 4 blocks), the overall structure is the same. For saving resources we excluded the last version.

3.2. FPN121 and FPN169

As we previously explained, our main goal was to verify if adding feature pyramids can boost the baseline models, trivially, we called *FPN121* the model with DenseNet121 as the backbone and analogously for the *FPN169*. The FPN121 is depicted in Fig. 1.
The first five blocks are exactly the components of the DenseNet121, where dense blocks contain deeply stacked convolution layers with batch normalization gates and the layers are tied each other through skip connections, transition blocks are served to reduce the input dimension including maxpooling layers. FPN is implemented by *top-down* pathway and *lateral connections*, the feedforward computation of the backbone is called *bottom-up* pathway. The top-down pathway is used because of the idea of enhancing the low-level feature by summing upsampled high-level features that are semantically stronger. We point out that before upsampling a 1x1 convolutional layer is needed to reduce the number of filters. Another observation is that we excluded the first block of the backbone model since we believe that it returns too low-level features that wouldn't help to improve the final results, additionally, the output's resolution is too high and it would increase significantly the overall computation time. Finally, we applied a Global Average Pooling layer [5] (GAP) to vectorize the 3-D feature data and then we immediately connect the vector to the output 1x1 fully connected layer to predict the counting numbers. We noticed that having a deep last stage MLP didn't contribute to increase the performance, instead, it made training more difficult.

4. Datasets

In this section we briefly describe each dataset we employed in this work and for each of them we show some sample images.

4.1. Mall dataset

The dataset is composed by 480x640 RGB images of frames in a video and the ground truth numbers of pedestrians are given in a csv files. The images come from a webcam placed in a fixed position in a mall and these data are freely available in Kaggle [1].
Data distribution histogram is drawn in Fig. 3. We see that the number of samples is small with respect to a usual machine learning dataset and the data are not equally distributed, meaning that there are few frequent target values along with many other values, these make this prediction task quite difficult.

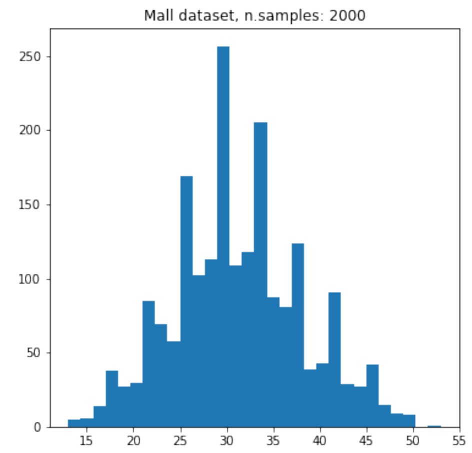


Figure 3: Mall dataset histogram.

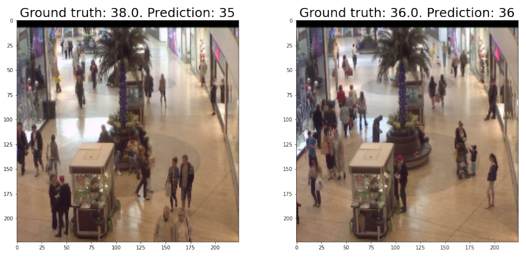
4.2. ShanghaiTech dataset

This large-scale crowd counting dataset is introduced in [13], it contains 1198 annotated images, with a total of 330,165 people with centers of their heads annotated. This dataset consists of two parts: there are 482 images in Part A which are randomly crawled from the Internet, and 716 images in Part B which are taken from the busy streets of metropolitan areas in Shanghai. Both Part A and Part B are divided into training and testing: 300 images of Part A are used for training and the remaining 182 images for testing;, and 400 images of Part B are for training and 316 for testing. We report in Fig. 4 and Fig. 5 the data distribution histograms. We can see that both parts are challenging dataset, especially the Part A and we notice this also from image samples where for the Part B the crowds are relatively sparse.

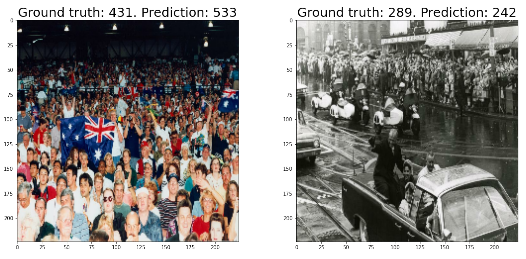
4.3. UCF-CC-50

This is another very challenging dataset containing images of extremely dense crowds and only 50 samples. So, not only the images are very dense but also the amount of data available is scarce. Again, we show the histogram in

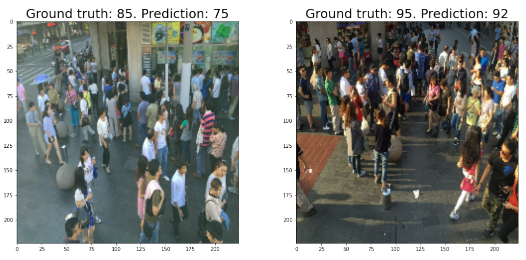
[1]https://www.kaggle.com/fmena14/crowd-counting


Ground truth: 38.0. Prediction: 35
Ground truth: 36.0. Prediction: 36

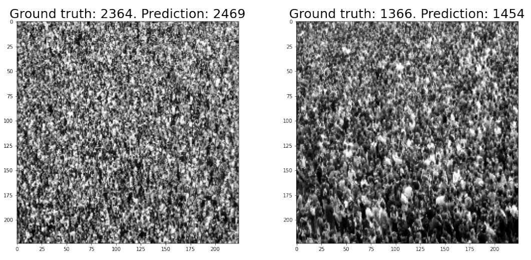
(a) **Mall dataset**


Ground truth: 431. Prediction: 533
Ground truth: 289. Prediction: 242

(b) **ShanghaiTech Part A dataset**


Ground truth: 85. Prediction: 75
Ground truth: 95. Prediction: 92

(c) **ShanghaiTech Part B dataset**


Ground truth: 2364. Prediction: 2469
Ground truth: 1366. Prediction: 1454

(d) **UCF-CC-50 dataset**

Figure 2: Some sample images for each dataset, the estimations are made by the FPN121.

Fig. 6. Also this dataset can be freely downloaded [2].

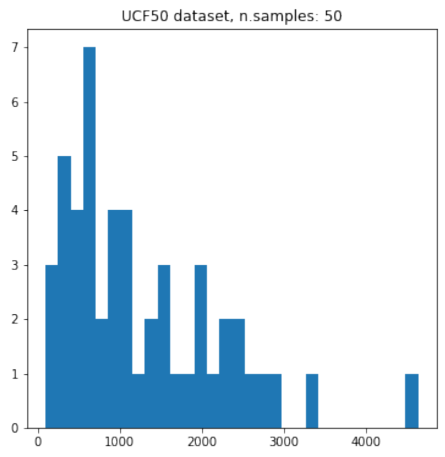


Figure 6: Mall dataset histogram.

5. Training and implementation details

5.1. Data preprocessing and augmentation

Mall dataset:

The original image resolution is too high, so we reduced the dimension to 224x224x3 that is the default input size of Imagenet. Then, even if the data available is insufficient, we preferred to split the full dataset in a training and a validation set (1580:420), the validation set is used to check during the model training whether the models overfit the data. Since the images are frames of a video recorded by a camera installed in a fixed place, the only useful data augmentation to apply is random horizontal flip in addition to the usual normalization operation on images.

[2]https://www.crcv.ucf.edu/data/ucf-cc-50/

ShanghaiTech dataset:

We resized the images and applied the normalization operation, along with random horizontal flip we also add random contrast image augmentation that we found helpful to mitigate overfitting. No dataset split procedure is needed.

UCF-CC-50 dataset:

For this dataset we applied the same image augmentation methods as in the previous case and since this is a tiny dataset we couldn't further split the dataset, but we still needed to evaluate the models. So, we simply duplicated the dataset and we tested the models on this no augmented dataset.

5.2. Training

We trained our models by minimizing the *mean square errors* (MSEs) and we evaluated them computing *mean ab-*

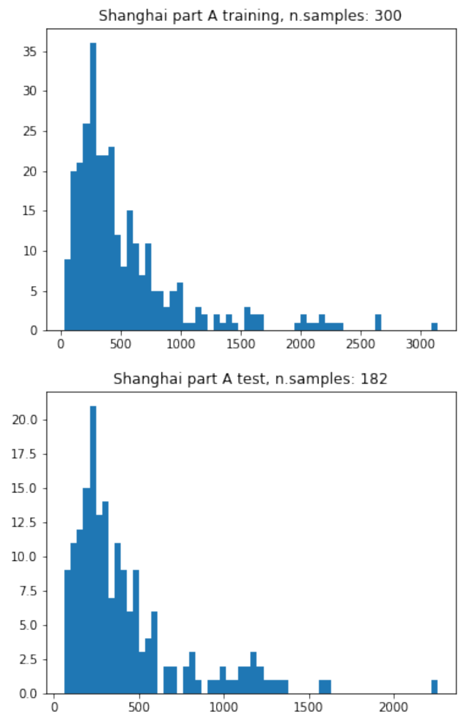


Figure 4: ShanghaiTech Part A.

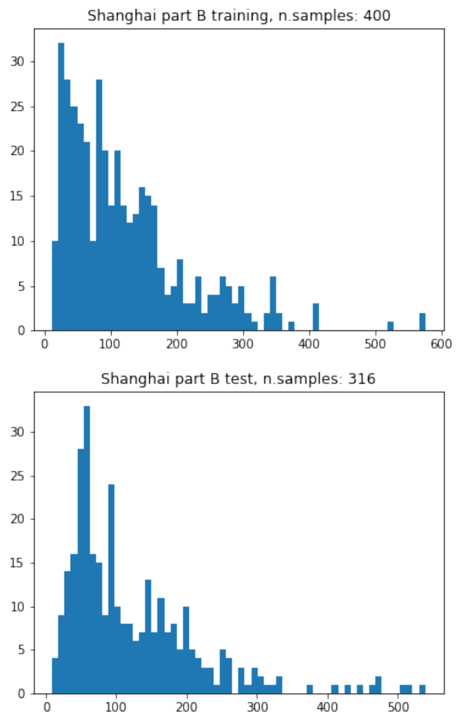


Figure 5: ShanghaiTech Part B.

solute errors (MAEs):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) + \lambda\|w\|_2,$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i)|,$$

where N is the total number of samples, y_i is the predicted number of the $i-$th sample and $\hat{y}_i$ is the ground truth. The last part of the loss is the regularization term of the last fully connected layer.
We tried different optimization algorithm such as the basic SGD with momentum or RMSprop but we found that the Adam, in this context, is the best choice.

5.3. Model implementation

We strived to set a learning rate scheduler as effective as possible and at the end either simple reduce on plateau or stepwise learning rate were applied.
We highlight that we didn't insert dropout layers into the networks because of the nature of this problem; some images are extremely dense, drop out pixels would be disastrous. Indeed, we verified practically that adding the latter components worsened the generalization performance of each model. So, we only relied on the L_2 regularization to keep away our models from overfitting trap.
Finally, we saved the best model parameters according to the lowest MAE and we evaluate the models reloading those parameters.
Transfer learning was implemented, specifically, since images' resolution coincide with those contained in the Imagenet, we initialized the weights equal to the pretrained model weights.
All experiments were conducted using **Python3** along with its popular machine learning libraries and **TensorFlow** framework, we run the codes on a 16 GB GPU, 16 GB RAM and quad-core CPU machine offered by Kaggle for free.

6. Results

The amount of data of each dataset are relatively small for a typical deep learning application, in fact they are mainly exploited by either classic computer vision algorithms or machine learning models that aimed also to produce density maps as we have already mentioned. Thus, what we report as related work results are for a rough comparative purpose, additionally, there are no clear descriptions of data preprocessing. However, our main focus is always on comparison between base models and enhanced models.

6.1. Results on Mall dataset

Model	MAE (full dataset)	N. parameters
DenseNet121	0.80	∼7.2M
DenseNet169	0.81	∼12.5M
MCNN [9]	4.74	N/A
ST-CNN [9]	4.03	N/A
FPN121	0.58	∼7.2M
FPN169	0.46	∼13.2M

Table 1: Results on Mall dataset, the models were evaluated on the full dataset without data augmentation.

As we can see from the Tab. 1 the feature pyramid models outperform their starting models with a little increase of the initial number of parameters, we observe during the training that the FPNs required less computation time for a training step, this is due to the downsampling operation we performed. Importantly, we note from the training plot in Fig. 7a that FPNs converged much faster and more stable (less variation in the loss) than the original models, this implying that feature pyramids are really capable to capture features of different scale in the images and this confirms what we intuited at the beginning.
Finally, we gained a very good results with respect to what we find in literature.

6.2. Results on ShanghaiTech dataset

We report the scores in Tab. 2. First of all we see that all models overfitted the data and whereas in the less tough task (Part B) the FPNs outperformed the baseline models in the other case the best network is the smallest one (DenseNet121). Nevertheless, the new version of the models converged faster (Fig. 7b and Fig. 7c) and gained less prediction errors on the training set meaning that we may need more sophisticated data argumentation techniques, but yet dropout layers cannot be used as we have already discussed. Even with different weight decay values the performance were poor.

Part B:		
Model	Training MAE	Test MAE
DenseNet121	8.9	23.2
DenseNet169	9.6	23.4
MCNN	N/A	26.4
SCRNet	N/A	10.6
FPN121	8.3	21.9
FPN169	8.7	22.6
Part A:		
DenseNet121	67.6	111.1
DenseNet169	57.5	130.5
MCNN	N/A	110.2
SCRNet	N/A	68.2
FPN121	48.4	131.8
FPN169	61.9	131.8

Table 2: Results on ShanghaiTech dataset, for the external scores we have no clues about how the authors evaluated the models; they might use full resolution.

6.3. Results on UCF-CC-50

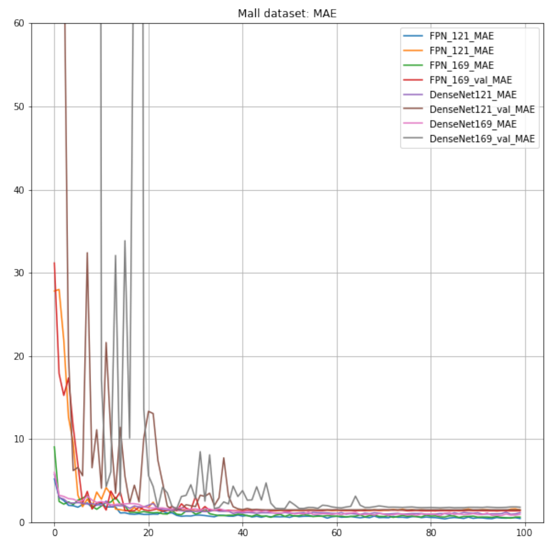
Again, the FPNs outperformed the others and the training convergence was accelerated by feature pyramids as we can see from Tab. 3 and Fig. 7d. We remark that, for this problem, larger models performed poorly, this is not so surprising because of the size of the dataset.

Model	Training MAE	Test MAE
DenseNet121	431.4	514.9
DenseNet169	264.9	519.0
MCNN	N/A	377.65
SCRNet	N/A	266.15
FPN121	155.1	233.6
FPN169	117.5	286.7

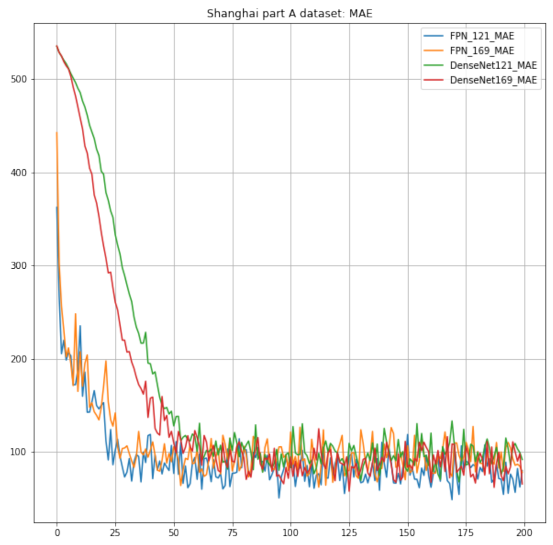
Table 3: Results on UCF-CC-50 dataset, we recall that the test set was simply a copy of the training set without data augmentation.

7. Conclusion

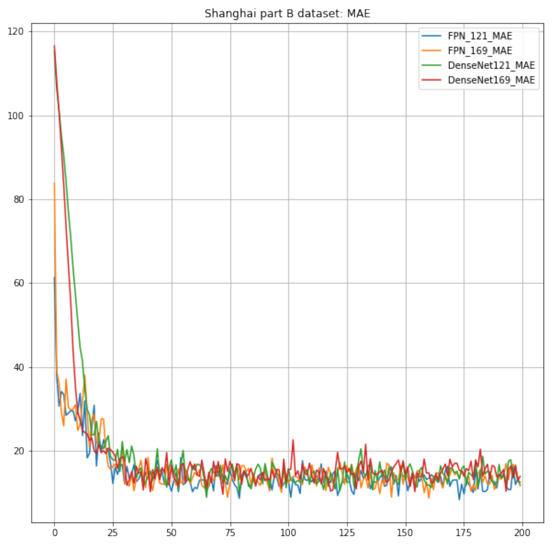
In this work we conducted crowd counting experiments on some public available datasets, we proposed some promising deep learning networks improving an existing model architecture (DenseNet). Although we had limited resources we were able to achieve outstanding results with respect to the state-of-the-art accuracies. We believe that it is possible to surpass the previous models by increasing the image resolution, that may be crucial in this context, creating more specific data augmentation operation for model training and doing extensive hyperparameter search.
Finally, since FPNs has the advantage of short training and inference time as they can be also implemented to perform object detection tasks, it would be interesting to test them in a real-time counting scenario.



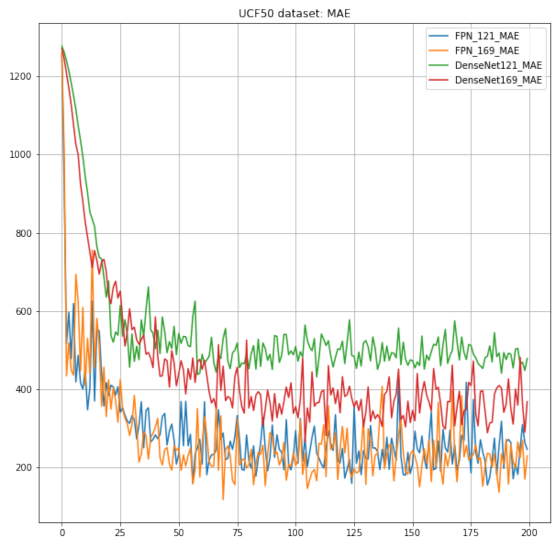
(a) **Mall dataset training MAE plot.**



(b) **ShanghaiTech Part A training MAE plot.**



(c) **ShanghaiTech Part B training MAE plot.**



(d) **UCF-CC-50 training MAE plot.**

Figure 7: Training plots.

References

[1] Guangshuai Gao, Junyu Gao, Qingjie Liu, Qi Wang, and Yunhong Wang. Cnn-based density estimation and crowd counting: A survey. *arXiv preprint arXiv:2003.12783*, 2020.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer*

vision and pattern recognition, pages 4700–4708, 2017.

[4] 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.

[5] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[7] W. Liu, D. Lian W. Luo, and S. Gao. Future frame prediction for anomaly detection – a new baseline. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] Chen Change Loy, Ke Chen, Shaogang Gong, and Tao Xiang. Crowd counting and profiling: Methodology and evaluation. In *Modeling, simulation and visual analysis of crowds*, pages 347–382. Springer, 2013.

[9] Yunqi Miao, Jungong Han, Yongsheng Gao, and Baochang Zhang. St-cnn: Spatial-temporal convolutional neural network for crowd counting in videos. *Pattern Recognition Letters*, 125:113–118, 2019.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[12] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[13] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016.