CountNet: Using supervised deep learning to count people in images

Juan Da Silva and Jeremias Traub

August 12, 2020

Contents

1 Introduction

1.1 Crowd Counting

Over the past few decades, an increasing number of research communities, have considered the problem of object counting as their main research direction. As a consequence, many works have been published to count the number of objects in images or videos across wide variety of domains such as crowding counting, cell microscopy, animals, vehicles, leaves and environment survey. In all these domains, crowd counting is of paramount importance, and it is crucial to building a more high-level cognitive ability in some crowd scenarios, such as crowd analysis and video surveillance. The increasing growth of the world's population and subsequent urbanization results in a rapid crowd gathering in many scenarios such as parades, concerts and stadiums. In these scenarios, crowd counting plays an indispensable role for social safety and control management.

Considering the specific importance of crowd counting aforementioned, we have attempted to design a robust model to address the problem of crowd counting.

1.2 Traditional Approaches

There are three main types of counting methods. *Detection-based* crowd counting is an approach to directly detect each of the target objects in a given image. The counts of targets in an image are automatically given as a byproduct of detection results. However, highly occluded objects and objects in drastically different scales make detection much more challenging. These issues make detection-based approaches infeasible in dense crowd scenes.

Regression-based methods directly map input crowd images to scalar values of counts, hence bypassing explicit detection tasks. Particularly, a mapping between image features and the crowd count is learned. Typically the extracted features are used to generate low-level information, which is learned by a regression model. However, regression based approaches mostly ignore the spatial information in the crowd images.

Density-based crowd counting preserves both the count and spatial distribution of the crowd. In an object density map, the integral over any subregion is the number of objects within the corresponding region in the image.

1.3 CNN-based Density Estimation

Convolutional Neural Networks (CNNs) have proven to be a powerful tool for estimating density maps from an input image. Extisting approaches include for example adapted U-Nets [10, 6] or multi-column networks [11]. The latter leverages a multi-column architecture using convolution filters of different sizes in each column to address perspective issues.

2 CountNet

We developed the *CountNet* model, a convolutional neural network which, given an input image, generates a density map of the same resolution as the input. Mostly, inspiration was drawn from the W-Net [10] and Ted-Net [6].

The whole CountNet project can be found on Git under `https://github.com/jeremiastraub/CountNet`.

2.1 Architecture

The CountNet model is based on the U-Net architecture (Figure 1) [8]. The U-Net architecture has an overall encoder-decoder structure with layers on different depth-levels between which 2×2 Max-Pooling (on the down-path) and Upsampling (on the up-path) is done. On each depth-level, encoder blocks extract features which are then concatenated and processed by the decoder blocks. Each decoder receives the features from the next lower depth-level concatenated with the features from the encoder of the same depth-level. The latter are processed as they are, i.e., we choose not to include skip-modules.

The CountNet encoder blocks consist of two subsequent 3×3 convolution layers, each followed by a `ReLU` activation. As decoder block we use two subsequent multi-scale inception modules (Figure 2), each followed by a `ReLU` activation. The inception modules are used to address the problem that different scales need to be taken into account, mainly due to persons at various image-depths. In the second inception module we use a dilation of 2 in the last 3×3 convolution of each branch to increase the receptive field size. In the following one decoder block as a whole is referred to as inception block. A final 1×1 convolution layer creates a single-channel density map from the output of the last decoder module.

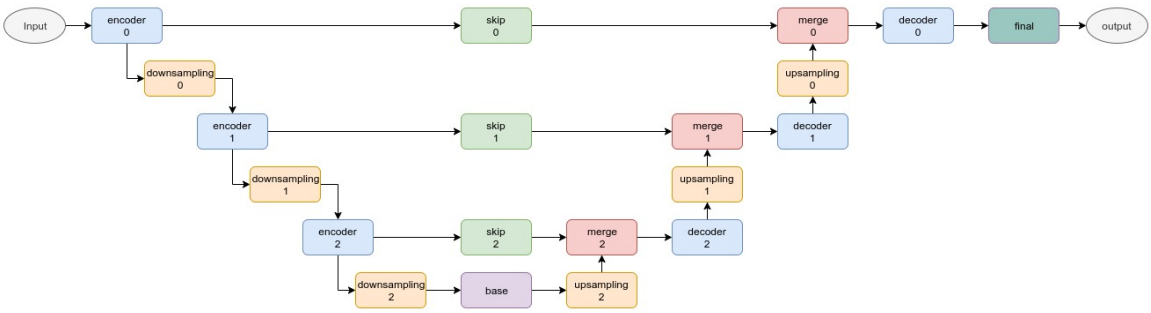


Figure 1: The U-Net architecture [8] on which the CountNet architecture is based. One can think of the vertical direction representing the *semantic* path, and the horizontal direction representing the *local* path. In the CountNet model, we use Max-Pooling as Downsampling module and simple Upsampling on the up-path (orange), common convolution layers as encoders/base and inception blocks as decoders (blue/purple), Identity transformations as skip-modules (green), concatenation along channels as merge modules (red) and a single convolution layer as final module (dark green). Image from: https://github.com/imagirom/ConfNets

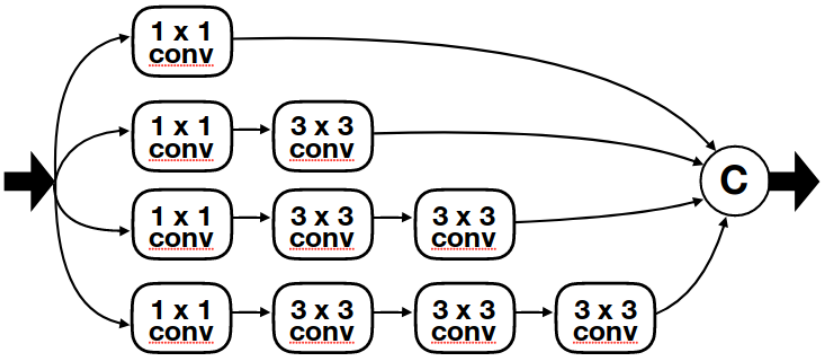


Figure 2: Inception module for extracting features from multiple local scales [9]. Convolutions of different size capture process features from different scales "in parallel". The features extracted on the 1×1, 3×3, 5×5, and 7×7 branch are then concatenated. The larger convolutions are broken down to chained 3×3 convolutions for computational efficiency.

3 Training

This section first contains a description of the datasets used and of the generation of the ground-truth density maps. Section 3.3 explains the training setup and configuration of the preprocessing.

3.1 Datasets

With the blooming development of crowd counting, numerous datasets have been introduced, which can motivate many more algorithms to cater to various challenges such as scale variations, background clutter in the surveillance video and changeable environment, illumination variation in the wild.

`Mall` dataset [1, 2, 7, 3]: This dataset contains 2000 frames of a single video camera showing pedestrians in a mall. It has diverse illumination conditions and crowd densities. Moreover, the scene contained in the dataset has severe perspective distortion. The dataset also presents severe occlusions. The video frames are of size 640×480 and contain 6000 instances of labeled pedestrians in total. We split the dataset into training and test set of size 1600 and 400, respectively.

`UCF` (CC 50) dataset [5]: This dataset includes a wide range of densities and diverse scenes with varying perspective distortion. It contains a total of 50 images with an average of 1280 individuals per image.

`Shanghai Tech` dataset [11]: This dataset consists of 1198 images with 330165 annotated heads. The dataset successfully attempts to create a challenging dataset with diverse scene types and varying density levels.

3.2 Ground-truth Generation

For the datasets that we used, the image data comes with point annotations marking the peoples' heads. In order to train the model on a pixel-level we first need to transform the annotations to a (smooth) density map. For that, we stack density maps for single annotation points for which the annotation points are each convolved with a Gaussian kernel G_{σ_i}. The resulting density map can be described by

$$F(x) = \sum_{i}^{N} \delta(x - x_i) \cdot G_{\sigma_i}(x)\,, \quad \text{with } \sigma_i = 0.45 \cdot \bar{d}_i \tag{1}$$

with $\bar{d}_i$ being the average distance to the 3 next nearest neighbors. The nearest neighbor search is done using a KD-tree. With each standard deviation being calculated based on the nearest neighbor distance, the width of the density peaks is adjusted towards the actual head size.

3.3 Training setup

For training we use only images of size 128×128. For that, we downscale the input images and then crop out a random patch of size 128×128[1]. Downscaling is done by applying a uniform filter first and then subsampling, in order to avoid aliasing.

The CountNet model (see Section 2) is trained on the pixel-wise MSE loss

$$\text{MSELoss} = \sum_{i,j} \big(\text{prediction}[i,j] - \text{groundtruth}[i,j]\big)^2 \tag{2}$$

with i, j running along the two image dimensions. If not specified explicitly, we use the `Adam` optimizer with a learning rate of 0.001. For the training we used the Google Colab with GPU and around 12GB RAM available.

We aimed at designing the project such that the pipeline of configuring the datasets and training, starting a training run, and evaluating previous runs is easy to use and that results are reproducible. For example, we use the `YAML` language for most configurations, which makes it easy to adjust parameters and configurations are stored alongside with model output.

[1] The images are downscaled as much as possible to still crop out a patch of size 128×128.

4 Validation metrics

There are several ways to evaluate the performance between predicted estimations and ground truths. We used some universally-agreed and popularly adopted measures for crowd counting model evaluation. These evaluation metrics are divided according to different evaluation criteria into three categories: image-level for evaluating the counting performance, pixel-level for measuring the density map quality and point-level for assessing the precision of localization. For our project we only used metrics of the first two categories.

Image-level metrics:

The two most common used metrics are Mean Absolute Error (MAE) and Mean Square Error (RMSE), which are defined as follows:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|C_{I_i}^{pred} - C_{I_i}^{gt}\right| \tag{3}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|C_{I_i}^{pred} - C_{I_i}^{gt}\right|^2} \tag{4}$$

where N is the number of test images, $C_{I_i}^{pred}$ and $C_{I_i}^{gt}$ represent the prediction results and ground truth of the count, respectively. Roughly speaking, MAE determines the accuracy of the estimates, while RMSE indicates the robustness of the estimates

To also take local information into account, the Grid Average Mean Absolute Error (GAME) [4] can be used:

$$\text{GAME}(L) = \frac{1}{N}\sum_{n=1}^{N}\left(\sum_{l=1}^{4^L}\left|C_{I_i}^{pred} - C_{I_i}^{gt}\right|\right) \tag{5}$$

where 4^L denotes the division of the image into non-overlapping regions. The higher L, the more restrictive of the GAME metric will be. We used $L = 4$ for this metric.

Pixel-level metrics:

Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index are usually used to measure the quality of the generated density map. PSNR is essentially based on the error between corresponding pixels, in other words, error sensitivity. However, it does not take the human visual characteristics into account. Therefore, the evaluation results are sometimes inconsistent with people's subjective feelings.

In addition, SSIM measures the image similarity from three aspects: brightness, contrast and structure, which can be regarded as the multiplication of the three parts. The value of SSIM is in the range of $[0, 1]$.

Point-level metrics:

To evaluate the localization performance of the model, Average Precision and Average Recall are the two most common used metrics.

5 Results

In this section we present selected results. First, we analyze the effect of the Inception decoder (Section 5.1), then we compare different network depths (Section 5.2). Both sections are based on the `Mall` dataset. With Section 5.3 we extend it to other datasets.

5.1 Inception Decoder

As described in Section 2.1, we propose a multi-scale Inception block as decoder. In order to analyze its effect, we use a basic convolution decoder as reference. The reference decoder consists of two subsequent 3×3 convolutions, each of them followed by a `ReLU` activation. We train both model setups with a network depth of $d = 3$ on the `Mall` dataset (batch size: 8).

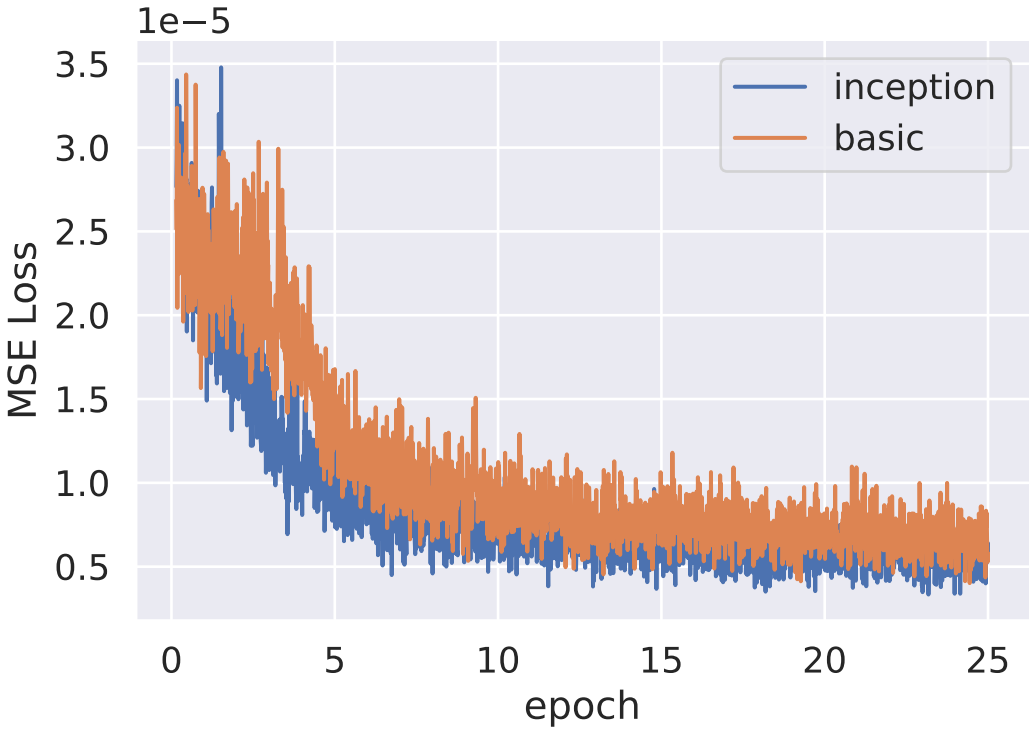


Figure 3: Comparing the loss developments over training epoch for the model setup with inception decoder and with basic convolution decoder. With the inception decoder slight training improvements are observed.

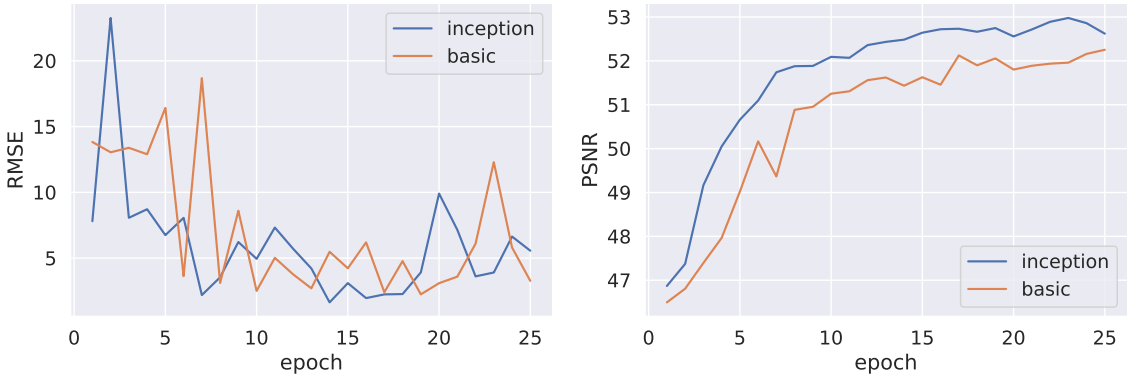
With the inception decoder, slightly lower losses are achieved and the model trains faster (Figure 3). In Table 1 the validation metrics evaluated after 25 epochs are shown. For both setups the difference of RMSE and MAE is small, the same holds for GAME and MAE. The former represents a measure for the stability of the resulting counts, the latter indicates local accuracy of the results (see Section 4).

While comparing the values for the basic and inception setup suggests that the basic model has higher counting accuracy (MAE, RMSE, GAME), looking at the variation of these metrics throughout the training reveals that the overall accuracy does not differ significantly (Figure 4a).

The opposite is the case for the PSNR. Its development during training suggests that there is consistent improvement of the PSNR using the inception decoder (Figure 4b). Hence, the inception decoder yields density maps of higher image quality.

Setup	MAE	RMSE	GAME	PSNR
inception	5.43	5.59	6.64	52.76
basic	2.61	3.11	4.09	52.27

Table 1: Validation metrics after 25 epochs for different model setups.



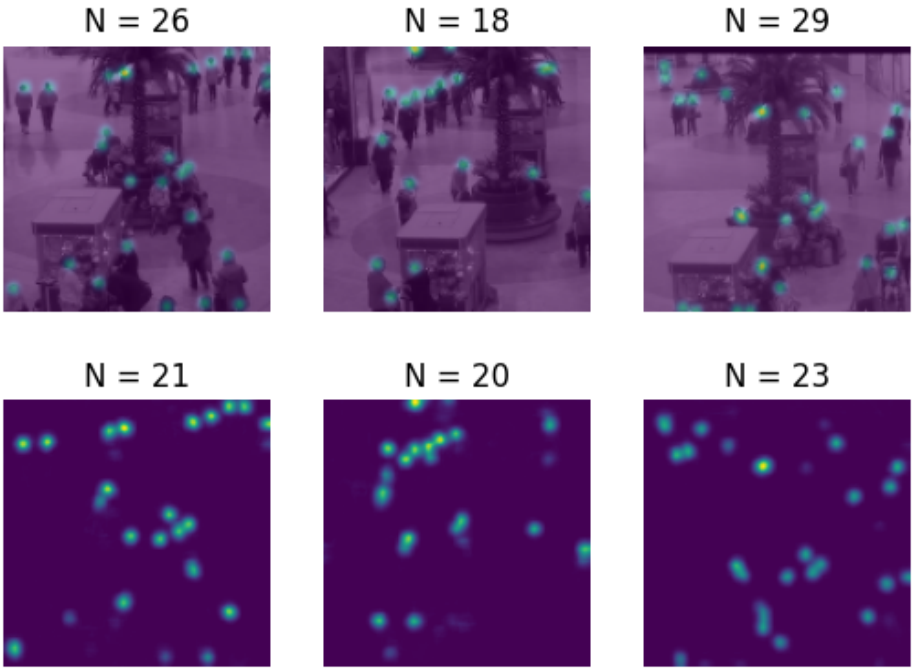
(a) RMSE over training epoch

(b) PSNR over training epoch

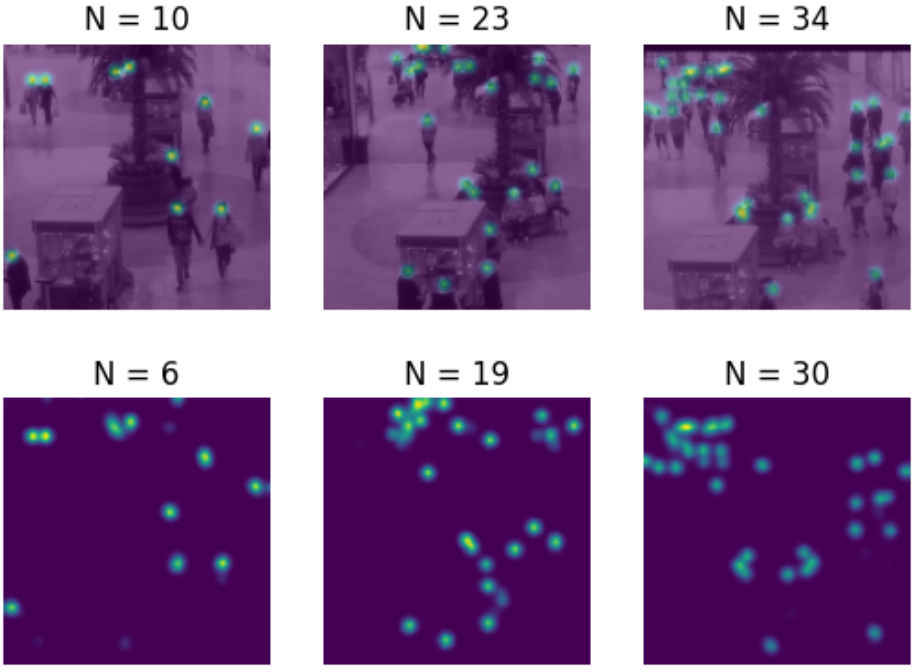
Figure 4: Comparing the RMSE (a) and PSNR (b) developments over training epoch for the model setup with inception decoder and with basic convolution decoder. The inception decoder does not improve the accuracy (RMSE) but it improves the quality of the generated density map (PSNR).

The improvement of image quality can also be observed when looking at the generated density maps. The density maps generated with the inception model (Figure 5b) are less noisy and the model appears to be more confident with the individual head annotations, in comparison to the basic model output (Figure 5a).

To analyze the counting accuracy in more detail, we plot the predicted counts against the true counts for both setups (Figure 6). Interestingly, the distributions look qualitatively different. For the basic setup, the predictions are accurate for lower counts, but deviate to lower values for higher counts. For the inception setup, the predictions appear to be systematically lower than the true counts, independent of the count. Noticeably, for higher counts the variation of the predicted counts stays lower than that of the basic setup, which increases for higher counts. We suppose that, on the one hand, the reduced variation might be due to the improved density map quality. This could be investigated further by analyzing the validation metrics for different counts. On the other hand, there appear to be systematic counting errors which might be rooted in the model architecture itself.

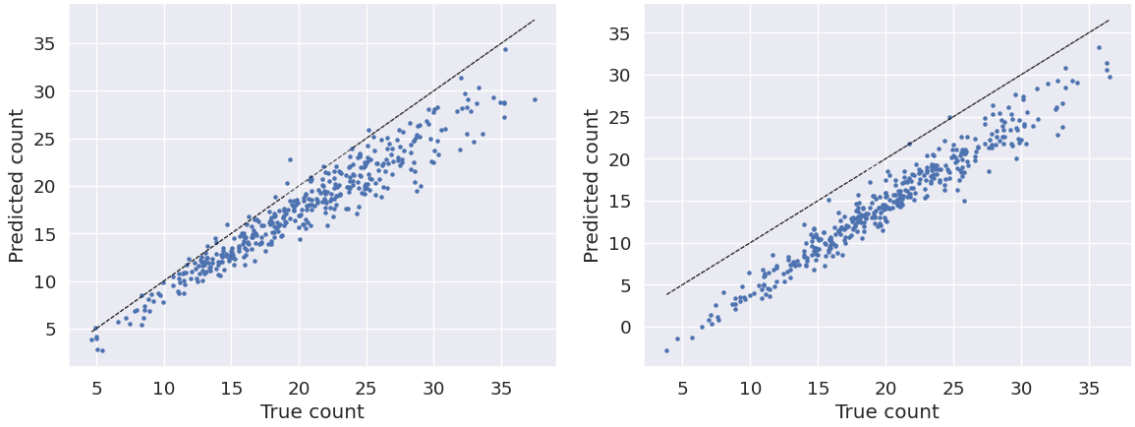


(a) basic



(b) inception

Figure 5: Example model output for the basic (a) and the inception (b) setup. The upper row shows the images with the overlayed ground-truth, the lower row shows the predicted density map.



(a) basic (b) inception

Figure 6: The count distributions for the basic (a) and the inception (b) setup. The predicted counts are systematically lower than the true counts.

5.2 Network Depth

To analyze the effects of the depth of the network, we compare the accuracy for different depths using the inception block as decoder. The network with depth $d = 1$ has $f = [128, 256]$ features, for higher depths we append layers with 64 features from the top.

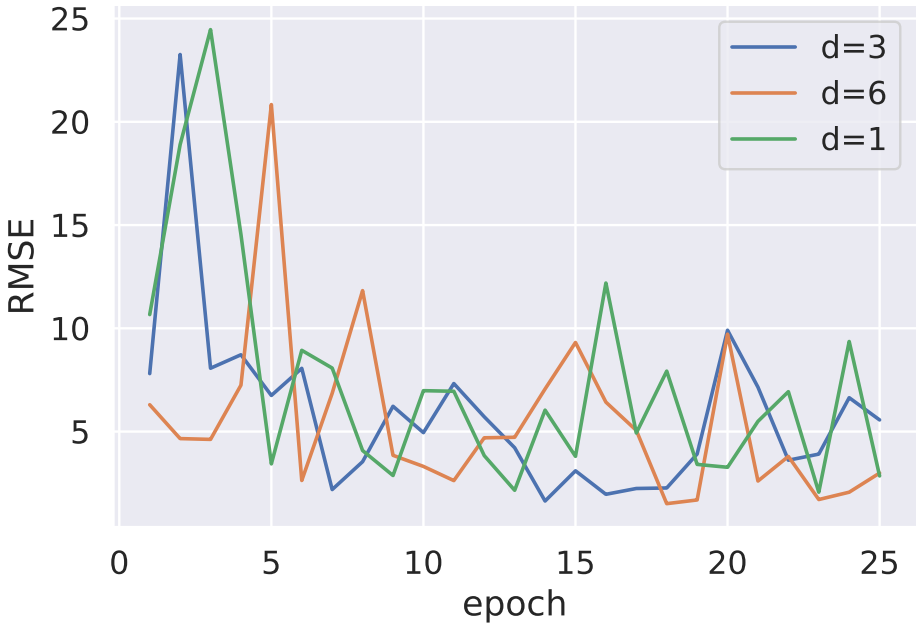


Figure 7: The RMSE development during training of inception models with various network depths d. No significant improvement in counting accuracy is achieved by increasing the network depth.

The RMSE shows no significant trend for the various depths (Figure 7). This indicates that increasing the network depth does not necessarily increase the counting accuracy. Increasing the depth increases the length of the *semantic path* of the U-Net. Compared to advanced semantic segmentation tasks, the "semantic difficulty" for the counting task is presumably rather low. Moreover, the inception decoder used for this comparison already provides some depth itself, which might just be *enough* for the model configuration, i.e. the used number of features, the image size, and in particular also the type of image data used (here: `Mall` dataset).

5.3 Dense Crowds

We attempted to extend the project to other types of training data, in particular to images of dense crowds. For that, we ran the training pipeline for the `UCF` and the `ShanghaiTech` datasets (see Section 3.1). However, the CountNet model could not be trained on these more difficult datasets. Partly, this could be because our image size of 128×128 is too small to resolve dense crowds. However, we suppose that the main reason lies in the model architecture. Next steps would be to investigate different architecture setups, for example to put more focus on the localization (horizontal) path of the network via skip-links. This provides much room for improvement but exceeds the time frame of our project.

6 Summary

We developed the CountNet model for counting people in images via CNN-based density map estimation. Training on the `Mall` dataset yields acceptable accuracy of the resulting count (see Section 5.1). Using multi-scale inception blocks as decoder yields higher quality density maps.

There is much room for improvement and extension, as the application of the model to more difficult data shows (see Section 5.3).

7 Contributions

All code for the CountNet project is available on Git under `https://github.com/jeremiastraub/CountNet`. We used the U-Net scaffold at `https://github.com/imagirom/ConfNets/blob/master/confnets/models/unet.py#L85` as a basis for our model implementation. Wherever code from other sources was used in this project, this is indicated within the code.

Both authors contributed equally to the planning of and literature research related to this project. All contributions to the code are traceable on the Git project page.

References

[1] C. Change Loy, S. Gong, and T. Xiang. From semi-supervised to transfer counting of crowds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2256–2263, 2013.

[2] K. Chen, S. Gong, T. Xiang, and C. Change Loy. Cumulative attribute space for age and crowd density estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2467–2474, 2013.

[3] K. Chen, C. C. Loy, S. Gong, and T. Xiang. Feature mining for localised crowd counting. In *BMVC*, volume 1, page 3, 2012.

[4] R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. Onoro-Rubio. Extremely overlapping vehicle counting. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 423–431. Springer, 2015.

[5] H. Idrees, I. Saleemi, C. Seibert, and M. Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2547–2554, 2013.

[6] X. Jiang, Z. Xiao, B. Zhang, X. Zhen, X. Cao, D. Doermann, and L. Shao. Crowd Counting and Density Estimation by Trellis Encoder-Decoder Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6126–6135, Long Beach, CA, USA, June 2019. IEEE.

[7] C. C. Loy, K. Chen, S. Gong, and T. Xiang. Crowd counting and profiling: Methodology and evaluation. In *Modeling, simulation and visual analysis of crowds*, pages 347–382. Springer, 2013.

[8] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567 [cs]*, Dec. 2015. arXiv: 1512.00567.

[10] V. K. Valloli and K. Mehta. W-Net: Reinforced U-Net for Density Map Estimation. *arXiv:1903.11249 [cs]*, Mar. 2019. arXiv: 1903.11249.

[11] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. 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.