Crowd counting on fixed camera images

Pierpaolo D'Odorico
pierpaolo.dodorico@studenti.unipd.it

Massimiliano Conte
massimiliano.conte@studenti.unipd.it

Abstract

In this work we compared different computer vision tecniques in order to estimate the number of people in a frame. The counting is performed on images captured from a fixed camera placed in a shopping mall. Some applications of this kind of counting on a static view are security and safety tasks, estimating the number of visitors on a mall for a/b testing purpose, planning spaces and services or verify compliance with covid-19 social distancing.

1. Introduction

The crowd counting problem received a lot of attention in recent years, due to its direct connection with crowd control and public safety. For this reason many techniques were recently proposed. Our idea is to compare two main tecniques in this fixed camera setting, one that is fast to implement and the other one more challenging, in order to verify if it is worth spending time for a more sophisticated solution. The first approach is to direcly estimate the number of people performing regression with a deep convolutional neural network, such as the VGG16 network [**simonyan2014very**]. We chose this very deep network since it is easy to handle for our purposes, unlike more complex architectures that have, for example, skip layer connections, and also because it is the base for the second technique. The second approach performs an undirect estimate of the number of people. First it is estimated the density of people in the image, then starting from the obtained density map the count is inferred. This second approach represents the base idea for the state of the art methods in crowd counting, where images could have a completely different number of people on differents enviroment and perspective. After implementing the two approaches on our problem, we found that a simple regression based on neural networks could perform as good as density based approach, probably exploiting the fixed background and because density estimation methods are well suited for dense datasets.

2. Related work

2.1. VGG16 net

We based our work on information contained in different papers about computer vision tasks and crowd counting. The first one is related to the base network of both approaches, the VGG16 net [**simonyan2014very**]. In this paper the authors investigated the effect of a deeper (with respect to previous architectures) convolutional neural network on the classification accuracy in the large-scale image recognition setting, specifically on the *imageNet* dataset [**deng2009imagenet**]. In this convolutional neural networks they used very small (3x3) convolution filters, which have shown a significant improvement on the prior state of the art configurations. They also pushed the depth to 16-19 weight layers (Fig.1). Those convolutional filters learned during *imageNet* classification task can be useful in our application, giving a meaningful feature extraction for finding people in images.

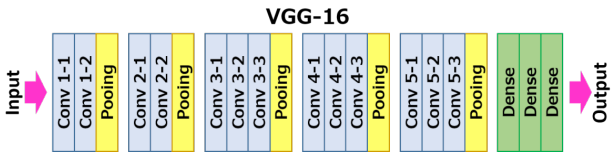


Figure 1: VGG16 deep CNN architecture.

2.2. Density based approach

The crowd counting based on density map estimation is a well known approach. One solution that is robust with respect to a variety of image properties is the *Multi-Column Convolutional Neural Network (MCCNN)* [**zhang2016single**]. The authors designed a system capable of detecting heads of different sizes, both in dense or sparse situations. Their work is first based on the generation of ground-truth density maps via geometry-adaptive kernels, for handling both dense and sparse images. Their architecture is designed in such a way that is able to detect heads of different sizes, in particular they built a neural network with three branches, one for each head size. They were consider small, medium and large heads (Fig.2). For the training they

pre trained separately each branch, and then the full network. The output of the network was designed and trained to be the estimation of the ground truth density maps. In order to train such a network, since are required ground truth density maps, a datatset with head annotations is a requisite. For this reason they also introduced *Shanghai-tech*, a new large scale crowd counting dataset.

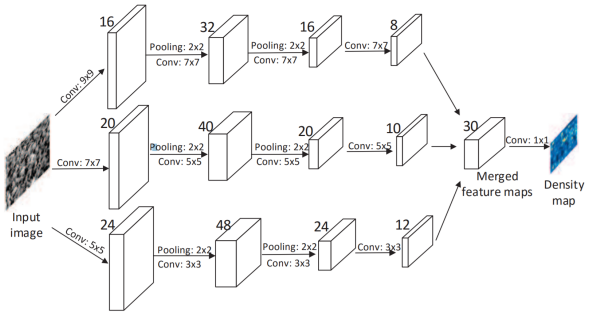


Figure 2: MCCNN architecture.

The models that we implemented are related to a more recent paper that simplifies the convolutional neural network of the previous work, by removing the multy-branch structure and going further with the deepness. They also experimented with the stride of convolutions, in an architecture they called *CSRNet* [**li2018csrnet**]. Designing a model which is automatically able to distinguish heads of different sizes achieved better results with respect to *MCCNN*. This architecture is basically the *VGG16* net with other convolutional layers on top of it, instead of the classifier we used in the simpler approach (Fig.3). For the training they fine tuned the network, starting from *ImageNet* learned weights for the $VGG16$ part and random normal initialization for the others. Since those methods [**zhang2016single**, **li2018csrnet**] don't use dense layers, they can work with images of any size.

3. Datasets

3.1. Mall dataset

The *Mall dataset* was introduced in 2012 in a paper about Localized Crowd Counting [**chen2012feature**]. It's composed by annotated RGB images of frames in a security camera video (Fig.4). The images have dimension 480 x 640 with 3 channels. The target variable is the human counting in each image. There are 2000 images in the dataset and we decided to split them using 1800 images for training and 200 test images. We randomly sampled 10% of data as test set becouse we think that it's a reasonable amount of data on which we can test our different models and different configurations.

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
conv3-512-2 conv3-512-2 conv3-512-2 conv3-256-2 conv3-128-2 conv3-64-2
conv1-1-1

Figure 3: Best CSRnet (kernel size - # of filters - stride).

Figure 4: Some images from Mall dataset.

The images shares a fixed background, but the height and width of human shapes can be very different between frames. This happens because people are walking along a mall corridor. For this reason some people are in foreground and other ones are in the background, and it is very likely to see small groups of people close to each other shopping together. Looking at the target variable distribution (Fig.5) we can see that the values are distributed around mean that equals to 31 humans in each image. This information can be useful for comparing the behaviour of our predictive models to the prediction performed by a dummy regressor that alwais predict the mean or the median of this distribution. We also notice that the dataset it's very sparse since it doesn't contain huge human crowds.

3.2. Shanghai tech dataset

This dataset was introduced in the *MCCNN* paper [**zhang2016single**]. This is a large-scale crowd counting dataset which contains 1198 annotated images, with a total of 330,165 people with centers of their heads annotated. It 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. The crowd density varies significantly between the two subsets, in particular the part A is

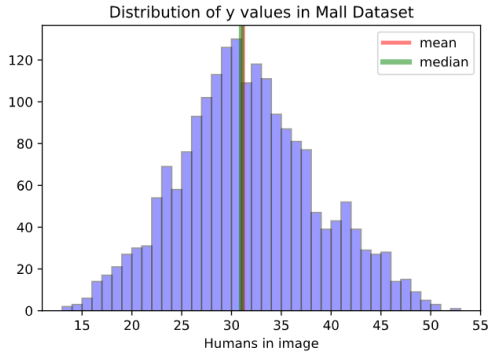
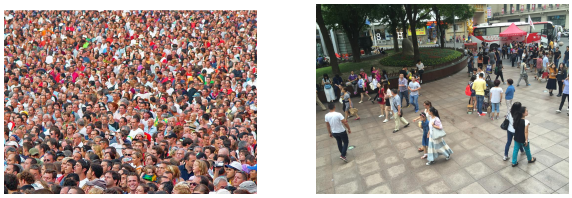


Figure 5: Mall dataset target variable distribution.

extremely dense, while part B is sparser, but it's still a dense dataset. 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 used Part B because is similar to the *mall dataset*, since it is sparser (Fig.6b).



(a) An image from Part A.

(b) An image from Part B.

Figure 6: Shanghai dataset.

4. Method

4.1. Regression based approach

4.1.1 VGG16 images preprocessing

In this first approach we want to directly estimate the number of people performing a regression using pre trained VGG16 deep convolutional neural network [**simonyan2014very**] as a feature extractor on *Mall dataset* images. Our idea is to keep the *ImageNet* learned weights freezed and to use them as a feature extractor for imeges. An important part of this approach is the image preprocessing. For reaching good results from pre trained VGG16 weights we need to preprocess *Mall dataset* images in the same way *ImageNet* images were processed before VGG16 network training. For this reason we started resizing the images in a squared 112 x 112 shape. In original VGG16 network images have a squared 224 x 224 shape, but we start from a halved height and halved width for faster training in our experiments. Then we will compare the best regression based approach models with *Mall dataset* images reshaped in 224 x 224 pixels for a more accurate result. In preprocessing we also convert the images from RGB to BGR, then we zero-center each color channel with respect to the *ImageNet* dataset without scaling (Fig.7). The result are squared images were peoples and objects in pictures are highlighted and the mall floor became white.

Figure 7: Mall dataset VGG16 preprocessing.

4.1.2 Adding layers to VGG16

On top of VGG16 deep network we want to add different combinations of dense layers and dropout for performing the regression. In this way we can train only the final layers of the network and we are able to perform more experiments due to the fact that training a limited number of layers is faster. We will train the final layers using SGD as described in [**li2018csrnet**]. Our idea is to train 4 different architectures with a 112 x 112 image shape. In first place we consider a shallow architecture, adding one dense layer with linear activation for performing regression. Than we create model that is the same to the first one but with a dropout layer for a better generalization. The third model is a deeper network, where dimension of layers is halved at each dense layer from 512 to 64 and than performing the regression on the final one. Than we repeat the same experiment with a dropout layer before each of those dense layers. The networks with a better performance will be trained with 224 x 224 input images for improving the performances.

4.2. Density based approach

4.2.1 Ground truth generation

Since we want the model to estimate the crowd density, we need ground truth density maps in order to perform supervised learning. A density map is a single channel image with positive values associated to each pixel. The generation of this image involve the head annotations: for each labeled image we have the coordinates of each head in that image. We can see each annotation as a sparse matrix, whose dimensions are the same as the image, with all zeros but one entry equals to one, corresponding to the pixel in the center of the head. As it is done in [**zhang2016single**],

we used geometry-adaptive kernels. The density map F is obtained via the following formula:

$$F(x) = \sum_{i=1}^{N} \delta(x - x_i) * G_{\sigma_i}(x) \text{ with } \sigma_i = \beta \bar{d}^i \quad (1)$$

Where:

- x represent a given pixel coordinates;
- N is the number of annotated heads;
- x_i is the i-th annotated head pixel coordinates;
- $\delta(\cdot)$ is the discrete version of the delta function: this function is zero on all possible points but in zero. In our case $\delta(0) = 1$ (in general we would have $\delta(0) = +\infty$ s.t. $\int_{Domain(x)} \delta(x)d(x) = 1$);
- $*$ is the convolution operator;
- G_σ is the 2-d gaussian filter;
- $\bar{d}^i$ is the mean euclidean distance between x_i and k neightbors.

As suggested in [**zhang2016single**], we used $\beta = 0.3$.

Figure 8: An example of ground truth density estimation.

4.2.2 CSRnet training

Our goal is to use this model on the *Mall dataset*, but we don't have annotation for those images. So we used *Shanghai-Tech* Part B dataset in order to pretrain the model. The first step was to build the architecture as described in [**li2018csrnet**], in particular the one that performed the best in their experiments (Figure 3). We frozen all the *VGG16* layers, with *ImageNet* pretrained weights, and train the other layers using SGD as described in [**li2018csrnet**], using *Mean Squared Error* pixel by pixel as loss function for the reconstruction of the density map.

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \sum_{x \in X^{(i)}} (\hat{F}^{(i)}(x) - F^{(i)}(x))^2 \quad (2)$$

Where $F^{(i)}$ is the i-th ground truth density map and $\hat{F}^{(i)}$ the estimation done by the model.
Then we fine tuned the whole neural network by train it again having unfrozen all the parameters, so we let the *VGG16* layers to adapt their filters in order to catch features that are more relevant for the crowd counting task. We checked the performance of the network by making predictions and evaluatng them using *Mean Absolut Error*.

$$\hat{y}^{(i)} = \sum_{x \in X^{(i)}} \hat{F}^{(i)}(x) \quad (3)$$

Basically the prediction is the sum over each pixel of the estimated density map.

$$MAE = \frac{1}{m} \sum_{i=1}^{m} |\hat{y}^{(i)} - y^{(i)}| \quad (4)$$

Since *VGG16* has 3 max pooling layers (Figure 1), we built the ground truth density maps such that its dimension would be $1/8$ of the original images, in order to make the predictions match the ground truth values.

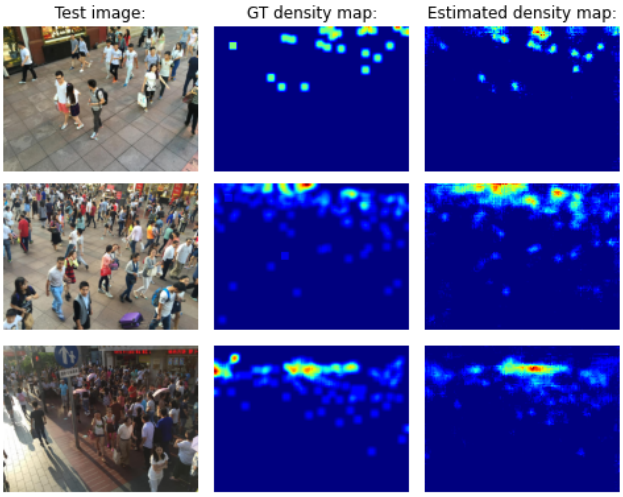


Figure 9: Test images, GT and CRSnet estimated density

As we can see from Figure 9, the estimation of the density maps provided by the trained *CSRnet* model seem reasonable.

4.2.3 More layers

Once we had the *CSRnet* trained on *Shanghai-Tech* Part B dataset we adapt this model to the *Mall dataset* in order to let it exploit the static background and the fixed perspective. We decided to use the fine tuned *CSRnet* as feature extractor and to build on top of it a regressor that estimates the number of people instead of integrating the density map. So we froze all the *CSRnet* weights and added some dense

layers (we experimented different configurations that will be explained in the next section), and that we trained the regressor with *Adam*, using as loss function the *MAE* on the prediction of the number of people. We could add dense layers since images on *Mall dataset* have fixed size. Then we also tried to fine tune the whole network.

5. Experiments

In this section we describe the experiments we run in each stage, and what's the best configuration that we found.

5.1. Regression based approach

In this section of the report we report the results of the experiments we performed with the simple regression based approach. In this way we are able to reach the best hyperparameter configuration for the trainable part of the network. We will evaluate the models with *MAE* metric (Equation (4)) computed on test set. We start comparing some models trained and evaluated on 122 x 122 pixel images. With this image reshape trick we are able to perform more experiments and to find a good hyperparameter tuning.

Architecture	Training procedure	*Mall dataset* MAE
112x112 images reshape +1 dense layer	*VGG16* frozen + Adam	3.25
112x112 images reshape + 1 dense layers + Dropout(0.1)	*VGG16* frozen + Adam	3.12
112x112 images reshape + 1 dense layers + Dropout(0.3)	*VGG16* frozen + Adam	2.82
112x112 images reshape + 5 dense layers	*VGG16* frozen + Adam	3.52
112x112 images reshape + 5 dense layers + Dropout(0.1)	*VGG16* frozen + Adam	3.27
112x112 images reshape + 5 dense layers + Dropout(0.3)	*VGG16* frozen + Adam	2.87

Table 1: Results on *Mall dataset* with 122x122 reshape
1 dense layer: 1 unit - linear activation;
5 dense layers: [512,256,128,64] units, 1 unit - linear;
Dropout is placed before each dense layer.

We reported the results in Table 1. We notice that the best dropout rate is 0.3 and that shallow networks and deeper ones show a similar behaviour on test data. In this way we are able to exploit more in detail the performances of the two best performing models, according to *MAE* metric, using a higher number of pixels in images. Models with 224 x 224 pixels show a little improvement in predicting people count (Table 1). As previously noticed the performance of the two networks on test data are similar.

Architecture	Training procedure	*Mall dataset* MAE
224x224 images reshape + 1 dense layers + Dropout(0.3)	*VGG16* frozen + Adam	2.53
224x224 images reshape + 5 dense layers + Dropout(0.3)	*VGG16* frozen + Adam	2.47

Table 2: Results on *Mall dataset* with 224x224 reshape
1 dense layer: 1 unit - linear activation;
5 dense layers: [512,256,128,64] units, 1 unit - linear;
Dropout is placed before each dense layer.

5.2. Density based approach

5.2.1 Shanghai Tech Part B

During the different phases of the *CSRnet* model training we checked the performances on *Shanghai-Tech* Part B dataset and on the unseen *Mall dataset*, in order to check whatever the model can be used for the specific unseen dataset. We report the *MAE* (Equation (4)) computed on test set.

CRSnet training procedure	*Shanghai-Tech* Part B MAE	*Mall dataset* MAE
VGG16 pre trained layers frozen and SGD on the new layers	40.17	14.34
Full previous trained neural network fine tuned using SGD	28.35	12.63

Table 3: CSRnet results.

In order to evaluate the capabilities of the model on the *Mall dataset* we need to know how a dummy regressor performs. Since we are using *MAE*, the best dummy regressor with respect to this metric is the median. A dummy regressor that always predicts the median has a *MAE* of around 6. So the *CSRnet* is performing very badly, and it needs to adapt to the specific characteristic of the dataset.

5.2.2 Mall dataset

In this section we performed several experiments starting for *CSRnet*. The reconstructions seem good (Figure 9), but the results on this dataset are bad. We started experimenting different network architectures for the new layers, we tried to use techniques such as dropout, resizing the starting images and we tried both the previous learned weights as feature extractor and to fine tune the whole architecture.

From Table 4 we can see how fine tuning the whole net does not work in this case. Having 2 dense layers perform slightly better, but it takes longer to train. Dropout is useful but too much bring distortion to the estimator. We tried also SGD but we didn't see significant differences.

Architecture	Training procedure	*Mall dataset* MAE
1 dense layer	Adam on full network	5.35
1 dense layer	*CSRnet* frozen + Adam	2.48
2 dense layers	*CSRnet* frozen + Adam	2.50
224x224 images rehape + 1 dense layer	*CSRnet* frozen + Adam	2.6
224x224 images rehape + 2 dense layer	*CSRnet* frozen + Adam	2.56
1 dense layer + Dropout(0.1)	*CSRnet* frozen + Adam	2.44
2 dense layers + Dropout(0.1)	*CSRnet* frozen + Adam	2.43
1 dense layer + Dropout(0.4)	*CSRnet* frozen + Adam	2.51
2 dense layers + Dropout(0.4)	*CSRnet* frozen + Adam	2.65

Table 4: Results on *Mall dataset*.
1 dense layer: 1 unit - relu activation;
2 dense layers: 1024 units - relu, 1 unit - relu;
Dropout is placed before each dense layer.

6. Conclusion

The best model we found is the 2 dense layer with dropout rate 0.2 trained on CSRnet frozen, with a *MAE* of 2.43. But the simplest approach of using a freezed VGG16 with a single dense regressor and dropout rate 0.1 performes a 2.53 *MAE*. In real world applications is hard to notice the prediction difference of the two models. Having a model that is able to count humans from images from a security camera with a Mean Absolute Error of 2.5 people can be useful for many applications. In such a fixed scenario does not help to use density maps approach, those methods are good on very dense images and performs similarly to simpler approaches on this setting, but they are harder to train, so it is not worth using them. For further developments purposes we can print some images with real human count and prediction (Fig.10, 11 and 12).

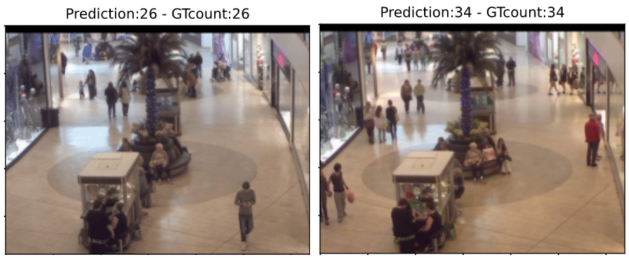


Figure 10: Mall dataset worst underestimated predictions.

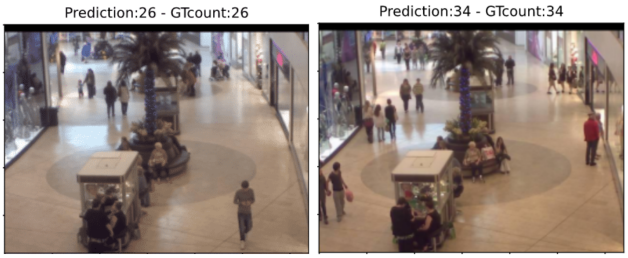


Figure 11: Mall dataset good predictions.

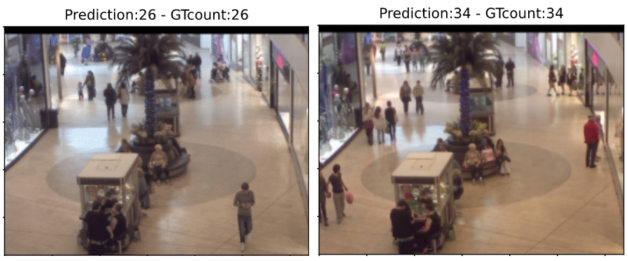


Figure 12: Mall dataset worst overestimated predictions.