Beyond Counting: Comparisons of Density Maps for Crowd Analysis Tasks - Counting, Detection, and Tracking

Di Kang, Zheng Ma, *Student Member, IEEE,* Antoni B. Chan *Member, IEEE,*

***Abstract*—For crowded scenes, the accuracy of object-based computer vision methods declines when the images are low-resolution and objects have severe occlusions. Taking counting methods for example, almost all the recent state-of-the-art counting methods bypass explicit detection and adopt regression-based methods to directly count the objects of interest. Among regression-based methods, density map estimation, where the number of objects inside a subregion is the integral of the density map over that subregion, is especially promising because it preserves spatial information, which makes it useful for both counting and localization (detection and tracking). With the power of deep convolutional neural networks (CNNs) the counting performance has improved steadily. The goal of this paper is to evaluate density maps generated by density estimation methods on a variety of crowd analysis tasks, including counting, detection, and tracking. Most existing CNN methods produce density maps with resolution that is smaller than the original images, due to the downsample strides in the convolution/pooling operations. To produce an original-resolution density map, we also evaluate a classical CNN that uses a sliding window regressor to predict the density for every pixel in the image. We also consider a fully convolutional (FCNN) adaptation, with skip connections from lower convolutional layers to compensate for loss in spatial information during upsampling. In our experiments, we found that the lower-resolution density maps sometimes have better counting performance. In contrast, the original-resolution density maps improved localization tasks, such as detection and tracking, compared to bilinear upsampling the lower-resolution density maps. Finally, we also propose several metrics for measuring the quality of a density map, and relate them to experiment results on counting and localization.**

***Index Terms*—Convolutional Neural Networks, crowd density map estimation, crowd counting, detection, tracking**

I. Introduction

Automatic analysis of crowded scenes in images and videos has applications in crowd management, traffic control, urban planning, and surveillance. The number of people and how they are spatially arranged are two useful measurements for understanding crowded scenes. However, counting, detection and tracking are still very challenging in low resolution surveillance videos, where people may only be a few pixels tall and occlusion frequently occurs – sometimes it can be very difficult even for a human expert. Detection-based methods can be used to count objects, but both their detection and counting performance will decrease as the scene becomes more crowded and the object size decreases. In contrast, regression-based counting methods [1–6] are more suitable for very crowded situations. However, most regression-based methods are designed only to solve the counting task, and cannot be used to localize the individual object in the scene. Methods that can simultaneously count and predict the spatial arrangement of the individuals will be more useful, since situations where many people are crowded into a small area are very different from those where the same number of people are evenly spread out.

Counting using *object density maps* has been shown to be effective at object counting [3, 4, 7–9]. In an object density map, the integral over any region is the number of objects within the corresponding region in the image. Similar to other regression-based methods, density-based methods bypass the difficulties caused when objects are severely occluded, by avoiding explicit detection, while also maintaining spatial information about the crowd, making them very effective at solving the object counting problem, especially in situations where objects have heavy inter-occlusion and appear in low resolution surveillance videos. Several recent state-of-the-art counting approaches [3, 4, 7, 8, 10] are based on density estimation. Furthermore, several works have explored how density maps can be directly used to localize individual objects [11], or used as a prior for improving traditional detection and tracking methods [12]. These works [11, 12] show the potential of applying density maps to other crowd analysis tasks.

The performance of density-based methods highly depends on the types of features used, and many methods use hand-crafted features [3–5] or separately-trained random-forest features [3, 4, 7, 9]. For very challenging tasks, e.g., very crowded scenes with perspective distortion, it is hard to choose which features to use. Indeed [5] used multi-source information to improve the counting performance and improve robustness.

One advantage of using deep neural networks is its ability to learn powerful feature representations. Recently, [8, 10, 13–15] introduced deep learning for estimating density maps for object counting. In [8], a CNN is alternatively trained to predict the crowd density map of an image patch, and predict the count within the image patch. The density map of the image is obtained by averaging density predictions from overlapping patches. In [10], a density map is predicted using three CNN columns with different filter sizes, which encourages the network to capture responses from objects at different scales. Both [8, 10] predict a reduced-resolution density map, due to the convolution/pooling stride in the CNNs. Density maps obtained by [8] also have problems of block artifacts and poor spatial compactness due to the way that the patches are merged together to form the density map. While these characteristics do not affect counting performance much, they do prevent individual objects from being localized well in the density map.

In this paper, we focus on a comparison of density map

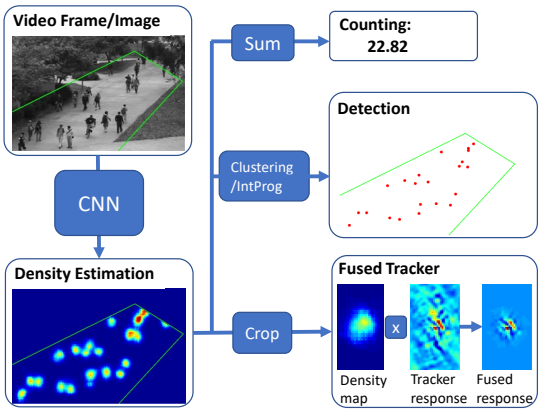


Figure 1: Crowd density maps for counting, detection, and tracking. A crowd count is obtained by summing the density map predictions within the ROI. Detections (the red dots in the figure) are obtained by GMM-weighted clustering or an Integer Programming formulation. Person trackers are improved by fusing (multiplying) the response of a traditional KCF tracker with the crowd density map.

estimation methods and their performance on counting and two localization tasks, detection and tracking (see Fig. 1). Since existing CNN-based methods normally generate a reduced-resolution density image, either to reduce parameters in the fully-connected layers or due to the downsample stride in the convolution/pooling layers, we also evaluate a full-resolution density map produced using a classic CNN and sliding-window to predict the density value for every pixel in the image (denoted as CNN-pixel). We also test a fully-convolutional adaptation of the CNN-pixel. To ensure its full-resolution output, the reduced resolution density map is upsampled to the original resolution using upsampling and convolution layers. Skip connections from the lower convolutional layers are used to compensate the spatial information. Various metrics are used to evaluate the quality of the density maps, in order to reveal why some density maps can perform better on localization tasks. We find that density maps that are spatially compact, well-localized, and temporally smooth are more suitable for detection and other future applications, such as object tracking.

The contributions of this paper are four-fold: 1) we present a comparison of existing CNN-based density estimation methods and two classic CNN-based methods on three crowd analysis tasks: counting, detection, and tracking; 2) we show that good detection results can be achieved using simple methods on high-resolution high-quality density maps; 3) we show that high-quality density maps can be used to improve the tracking performance of standard visual trackers in crowds; 4) we propose several metrics on density maps as indicators for good performance at localization tasks.

The remainder of the paper is organized as follows. Section II reviews related work in crowd counting, and Section III introduces our object counting method. Metrics for measuring the quality of various kinds of density maps are discussed in Section IV, and experimental results on counting, detecting, and tracking are presented in Section V.

II. Related Work

In this section we review crowd counting methods, and detection and tracking methods based on density maps.

A. Regression-based Counting

Perfectly detecting all the people or tracking every person in a video can solve the counting problem, and detection-based [16] and tracking-based [17, 18] counting methods have been proposed. However, their performances are often limited by low-resolution and severe occlusion. In contrast, regression-based counting methods directly map from the image features to the number of people, without explicit object detection. Such methods normally give better performance for crowded scenes by bypassing the hard detection problem. For regression-based methods, initial works are based on regressing from global image features to the whole image or input patch count [5, 19–22], discarding all the location information, or mapping local features to crowd blob count based on segmentation results [1, 2, 23]. These methods ignore the distribution information of the objects within the region, and hence cannot be used for object localization. Extracting suitable features is a crucial part of regression-based methods: [1] uses features related to segmentation, internal edges, and texture; [5] uses multiple sources (confidence of head detector, SIFT, frequency-domain analysis) because the performance of any single source is not robust enough, especially when the dataset contains perspective distortion and severe occlusion.

However, CNNs can be trained to extract suitable task-specific features automatically, and are inherently multi-source because they use many feature maps. Another advantage of the CNN approaches is that they do not rely on foreground-background segmentation (e.g., normally based on motion segmentation), and hence can better handle single images or stationary objects in a scene.

B. Density-based Counting

The concept of an *object density map*, where the integral (sum) over any subregion equals the number of objects in that region, was first proposed in [3]. The density values are estimated from low-level features, thus sharing the advantages of general regression-based methods, while also maintaining location information. [3] uses a linear model to predict a pixel's density value from extracted features, and proposed the Maximum Excess over Sub Array (MESA) distance, which is the maximum counting error over all rectangular subregions, as a learning objective function. [4] uses ridge regression (RR), instead of the computationally costly MESA, in their interactive counting system. Both [3] and [4] use random forest to extract features from several modalities, including the raw image, the difference image (with its previous frame), and the background-subtracted image.

[7, 9] used regression random forests, generated using the Frobenius norm as their criteria to obtain the best splits of their nodes. [7] uses a number of standard filter bank responses (Laplacian of Gaussian, Gaussian gradient magnitude and eigenvalues of the structure tensor at different scales), along with the raw image, as their input of the regression random forest. Other than the filter channels used in [7], [9] also uses the background subtraction result and the temporal derivative as their input of the regression random forest. Unlike [7], which directly regress the density patch, [9] regresses the vector label pointing at the location of the objects within

Methods	loss function	prediction	output resolution	feature/prediction method
MESA [3]	MESA (region)	pixel	full	random forest features, linear
Ridge regression (RR) [4]	per-pixel squared-error	pixel	full	random forest features, linear
Regression forest [7]	Frobenius norm	patch	full	regression random forests
COUNT forest [9]	Frobenius norm or entropy	patch	full	regression random forests
CNN-patch [8]	per-pixel squared-error	patch reshaped from FC	reduced	CNN
Hydra CNN [14]	per-pixel squared-error	patch reshaped from FC	reduced	CNN
CNN-boost [24]	per-pixel squared-error	patch reshaped from FC	reduced	CNN
cell-FCNN [13]	per-pixel squared-error	image from conv layer	full	FCNN
MCNN [10]	per-pixel squared-error	image from conv layer	reduced	FCNN
CNN-pixel	per-pixel squared-error	pixel	full	CNN
FCNN-skip	per-pixel squared-error	image from conv layer	full	FCNN

Table I: Comparison of methods for estimating object density maps: (top) using traditional features; (bottom) using deep learning.

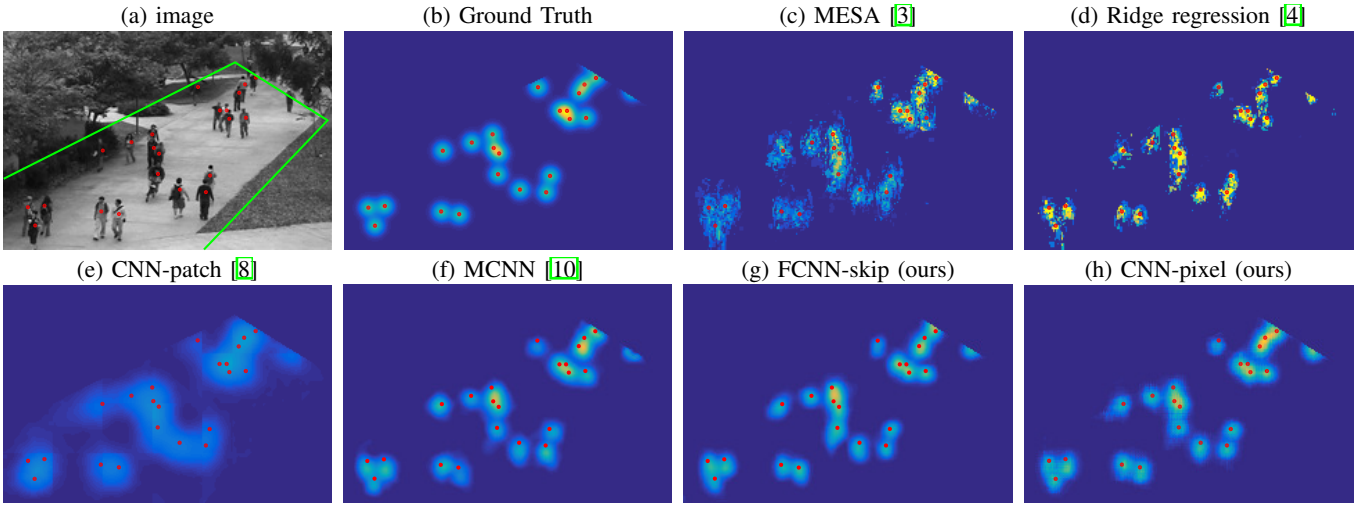


Figure 2: Comparison of different density map methods. All the density maps are in same color scale (a particular density value corresponds to the same color across images). The green line in the image shows the region of interest (ROI). The red dots are the ground-truth person annotations. The density maps of [8, 10] are resized to the original resolution via bicubic interpolation.

certain radius from the patch center, saving memory. The density patch is finally generated from predicted vector labels.

Among deep learning methods, CNN-patch [8], Hydra CNN [14] and CNN-boost [24] use the input patch to predict a patch of density values, which is reshaped from the output of fully connected layer. In contrast to [8], Hydra CNN extracts patches of different sizes and scales them to a fixed size before feeding them to each head of the Hydra CNN, while CNN-boost uses a second and/or a third CNN to predict the residual error of earlier predictions. [13] uses a classical FCNN to predict a density map for cell counting. In contrast, MCNN [10] is an FCNN with multiple feature-extraction columns, corresponding to different feature scales, whose output feature maps are concatenated together before feeding into later convolution layers. In contrast to Hydra CNN [14], the three columns in MCNN [10] use the same input patch but have different receptive field settings to encourage different columns to better capture objects of different sizes.

Density map estimation methods differ in their choice of training loss function and form of prediction, which result in different characteristics of the predicted density maps. Both the loss and the prediction can be either pixel-wise or region-wise. Table I summarizes the differences, and a visual comparison can be seen in Fig. 2. For the loss function, [3] uses a region-based loss consisting of the Maximum Excess over Sub Arrays (MESA) distance, which is specifically designed for the counting problem, and aims to keep good counting performance over all sub-regions. However, for MESA, per-pixel reconstruction of the density map is not the first priority so that [3] cannot well maintain the monotonicity around the peaks and spatial continuity in the predicted density maps (see Fig. 2), although it is partially preserved due to neighboring pixels being assigned to the same feature codeword and thus same density value. Most other methods [4, 7, 8, 10, 13, 14, 24] use a pixel-wise loss function, e.g., the squared error between the predicted density value and the ground-truth. While per-pixel loss does not optimize the counting error, it typically can yield good estimators of density maps for counting.

For density map prediction, traditional methods in [3, 4] choose a pixel-wise density prediction so as to obtain a full-resolution density map. The density map of the whole image is obtained by running the predictor over a sliding window in the image. In contrast, most deep learning-based methods choose a patch-wise or image-wise prediction to speed up the prediction [8, 10, 13, 14, 24]. Image-wise predictions using FCNNs [10, 13, 25] are especially fast since they reuse computations.

For patch-wise predictions, as in [8], patches of density maps are predicted for overlapping image patches. The density map of the whole image is obtained by placing the density patches at their image position, and then averaging pixel density values across overlapping patches. The averaging process overcomes the double-counting problem of using overlapping patches to some extent. However, due to the lack of context information around the borders of the image patch, the density predictions around the corners of the neighboring patches are not always consistent with (as good as) those of the central

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2837153, IEEE Transactions on Circuits and Systems for Video Technology

patch. The overlapping prediction and averaging operation can temper these artifacts, but also results in density maps that are overly smooth (e.g., see Fig. 2e).

The current CNNs using image- or patch-wise prediction normally only produce reduced-resolution density maps, due to either the convolution/pooling stride operation for FCNN-based methods, or to avoid very wide fully-connected layers for the patch-wise methods. Accurate counting does not necessarily require original-resolution density maps, and using reduced-resolution maps in [8, 10] can make the predictions faster, while still achieving good counting performance. On the other hand, accurate detection requires original resolution maps – upsampling the reduced-resolution maps, in conjunction with averaging overlapping patches, sometimes results in an overly spread-out density map that cannot localize individual object well. Considering these factors, our study will also consider full-resolution density maps produced with CNNs, in order to obtain a complete comparison of counting and localization tasks.

C. Detection and Tracking with Object Density Maps

Besides counting, [11, 12] have also explored using density maps for detection and tracking problems in crowded scenes. [11] performs detection on density maps by first obtaining local counts from sliding windows over the density map from [3], and then uses integer programming to recover the location of individual objects. In contrast to [11], our predicted density maps have clearer peaks, thus allowing for simpler methods for detection, such as weighted GMM clustering.

[12] uses density maps in a regularization term to improve standard detectors and tracking. In particular, a term is added to their objective function that encourages the density map generated from the detected locations to be similar to the predicted density map, so as to reduce the number of false positives and increase the recall. The density maps estimated in [12] are predicted from the detector score map, rather than image features, resulting in spread-out density maps. In contrast to [12], we show that, when the density maps are compact and focused around the people, a simple fusion strategy can be used to combine the density map and the response map of a visual tracker (e.g., kernel correlation filter).

III. Methodology

We consider two approaches for generating full-resolution density maps, as shown in Figure 3. The first approach uses a classic CNN regressor for pixel-wise density prediction, i.e., given an image patch, predict the density at the center pixel. The full-resolution density map is obtained using a sliding window to obtain density values for all pixels inside the ROI. Although pixel-wise prediction does not explicitly model the relationship between neighboring pixels, it still results in smooth density maps – the pooling operation in the CNN introduces translation invariance, and thus neighboring patches have similar features. Thus, the pixel-wise predictions using CNNs will tend to be smooth and can better maintain the monotonicity, which will benefit localization tasks, such as detection and tracking. In addition, due to the capability of CNNs to learn feature representations, density maps predicted by CNNs are less noisy and well localized around the objects, as compared to methods using handcrafted features (e.g., [4]).

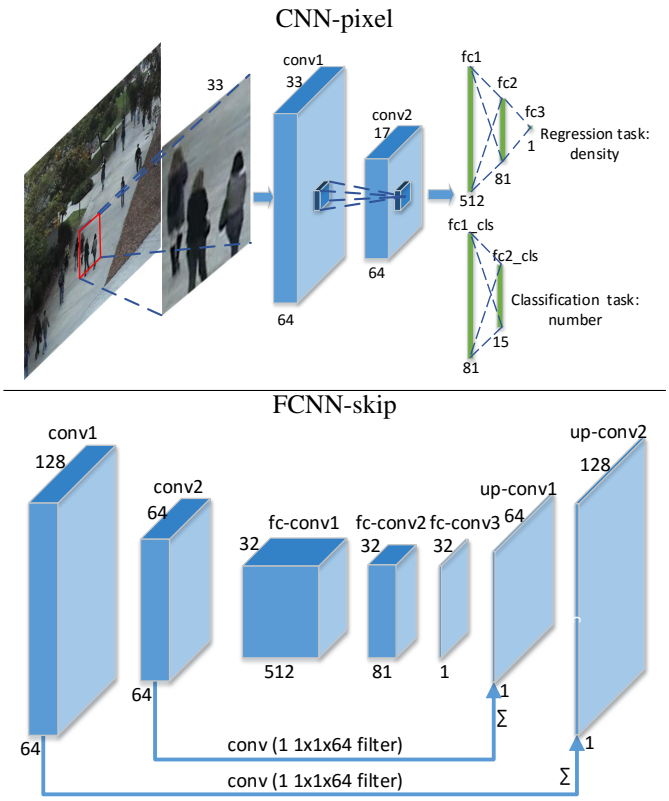


Figure 3: CNN architectures for full-resolution density maps: (CNN-pixel) pixel-wise prediction using CNN; (FCNN-skip) fully-convolutional NN with skip branches. Only layers with trainable weights are shown. A multi-task framework is used for CNN-pixel, consisting of: 1) predicting the density value at the center of the image patch; 2) predicting the number of people in the image patch. The two tasks share the same CNN feature extraction layers.

The second approach uses a fully-convolutional NN to perform image-wise prediction. Since the convolution/pooling stride operations in the lower-level convolutionl layers result in loss of spatial information (reduced resolution), subsequent upsampling and convolution layers are used to obtain a full-resolution density map. The FCNN also inherits the smoothness property of pixel-wise prediction, but is more efficient in the prediction stage since it reuses computations from overlapping regions of neighboring patches.

A. Pixel-wise Architecture

Our network architecture for pixel-wise prediction (denoted as CNN-pixel) is shown in Fig. 3 (top). The input for our network is 33×33 image patch, while the output is the density value at the center pixel of the input patch. The patch size is selected to be similar to the size of the largest person in the image. Image feature extraction consists of two convolution-pooling layers, followed by 3 fully-connected layers to predict the density value. A characteristic of the features extracted by CNNs is its hierarchical property: higher layers learn more abstract image features. On the contrary, crowd density is a mid-level feature, which reflects the count and spatial distribution of objects, and hence high-level abstract features are not necessary. Indeed, hand-crafted low-level and local

features work very well in previous regression-based counting methods [1–3, 23]. Hence, instead of a very deep architecture (e.g., [26–29]), we instead use an architecture that extracts mid-level features while keeping the model small and efficient.

For training, a ground-truth density map is generated from the "dot" annotations of people in the training images following [3]. For training image $\mathcal{I}_n$, the density map is obtained by convolving the annotation map with a Gaussian (see Fig. 2b),

$$D_n(p) = \sum_{P \in \mathcal{P}_n} \mathcal{N}(p; P, \sigma^2 I), \tag{1}$$

where p denotes a pixel location, and $\mathcal{P}_n$ is the set of annotated positions for image $\mathcal{I}_n$. $\mathcal{N}(p; P, \sigma^2 I)$ is a Gaussian density with mean P and isotropic covariance $\sigma^2 I$. For an image patch, its ground truth density value is the value of the ground-truth density map at the center of the patch. We use the squared Euclidean distance as the loss function,

$$\ell_{\text{density}}(d, \hat{d}) = (d - \hat{d})^2, \tag{2}$$

where $\{d, \hat{d}\}$ are the ground truth and predicted density values.

Similar to [8, 30, 31], we introduce an auxiliary task, which shares the same CNN feature extraction layers of the primary regression task, in order to guide the network to find good image features and to make the training of the regression task less sensitive to weight initialization and the learning rate. We choose a classification task as our auxiliary task because, empirically, classification is more robust to train than regression since it only needs to decide to which class the input belongs, rather than predicts an exact real number [32, 33]. Also, cross entropy loss for classification is less sensitive to outliers compared to L2 loss for regression [34]. The auxiliary task is a multi-class classification task where each class is the count of people within the image patch. The ground-truth count for this auxiliary task is the sum of the dot annotations within the region of the image patch. The categorical cross entropy is used as the loss function of the auxiliary task,

$$\ell_{\text{aux}}(p, \hat{p}) = \sum_i -p_i \log \hat{p}_i \tag{3}$$

where p_i is the true probability of class i (i.e., 1 if the true class is i, and 0 otherwise), and $\hat{p}_i$ is the predicted probability of class i. It should be noted that the classification loss ℓ_{aux} may not be a natural choice for measuring count prediction error, since it ignores the difference between the prediction and the true count. Nonetheless, this classification task is only used as an *auxiliary* task in the training stage. At test time, the count is estimated as the sum of the predicted density map. The regression and classification tasks are combined into a single weighted loss function for training,

$$\ell = \lambda_1 \ell_{\text{density}}(d, \hat{d}) + \lambda_2 \ell_{\text{aux}}(p, \hat{p}), \tag{4}$$

where weights $\lambda_1 = 100$ and $\lambda_2 = 1$ in our implementation.

Given a novel image, we obtain the density map by predicting the density values for all pixels using a sliding window over the image.

B. Fully Convolutional Architecture

Fully convolutional neural networks (FCNNs) [13, 25, 35, 36] have gained popularity for semantic image segmentation and other tasks requiring dense prediction because they can efficiently predict whole segmentation maps by reusing computations from overlapping patches. Efficiency can be further increased through up-sampling with a trainable up-sample filter. [25, 35] introduced "skip branches" to the FCNN, which add features from the lower convolutional layers to the up-sampled layers, in order to compensate for the loss of spatial information due to the stride in the convolution/pooling layers.

Our CNN-pixel is adapted to a fully-convolutional architecture, and includes skip-branches (see Fig. 3 bottom, denoted as FCNN-skip) as in [25]. In particular, the two pooling operation reduces the resolution of the density map by 4. Two up-sampling layers are then used to obtain the density map at the original resolution. Each up-sampling layer consists of two parts: each pixel is replicated to a 2×2 region, and then a trainable 3×3 convolutional layer is applied. The skip branches pass the feature maps from a lower convolution stage through a convolution layer, which is then added to the density map produced by the upsampling-convolution layer. The combination of features from different levels from a network has been shown to be helpful in better recovering lost spatial information [25, 35]. For semantic segmentation in [25], the skip connections help to preserve the fine details of the segments. For our counting task, by merging the low-level (but high resolution) features with the high-level (but low-resolution) features, the predicted high-resolution density map has more accurate per-pixel predictions than both pre-defined and learned interpolations. The skip-connections serve as a complementary part, which resembles residual learning, to refine the less accurate upsampled density map.

For FCNN training, we use both pixel-wise loss and patch-wise count loss,

$$\ell_{\text{pixel}} = \sum_{i,j} (d_{i,j} - \hat{d}_{i,j})^2, \tag{5}$$

$$\ell_{\text{count}} = \left(\sum_{i,j} d_{i,j} - \sum_{i,j} \hat{d}_{i,j} \right)^2, \tag{6}$$

where (i, j) index each pixel in one training patch. The pixel-wise loss ensures that the predicted density map is a per-pixel reconstruction of the ground-truth map, while the count loss tunes the network for the final target of counting. During training, we initialize the FCNN using a trained CNN-pixel model, and hence an auxiliary task is not needed. In the prediction stage, we give the whole image as input and predict the density map with the same resolution. Similar to CNN-pixel, there are no block artifacts in the predicted density map (e.g., see Fig. 2g and Fig. 10).

C. Detection from Density Maps

Detecting objects directly from density maps was first proposed in [11]. In [11], a sliding window is passed over the density map to calculate the object count within each window. [11] used the density maps produced by MESA [3]. Recovering the object locations is then a deconvolution problem, which is implemented as a 2D integer programming problem since there can only be a nonnegative integer number of objects at each location.

Besides integer programming, [11] also proposed several simple baseline methods: 1) using non-maximum suppression

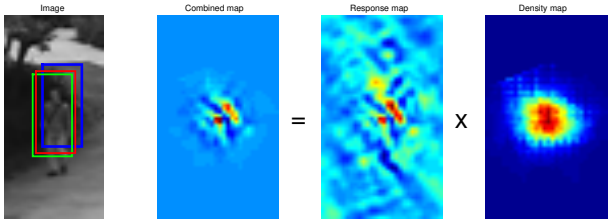


Figure 4: An example of tracking by fusing the kernel correlation filter (KCF) response map and the crowd density map. There are two peaks in the KCF response map, which causes the KCF tracker to drift (blue box). Fusing the density map and the response map downweights the erroneous peak, and thus prevents tracker drifiting (green box). The red box is ground truth bounding box.

to find local peaks in the density map; 2) using k-means clustering on the pixel locations with density value above a threshold (a simple form of crowd segmentation); 3) using Gaussian mixture model (GMM) clustering on the thresholded density map. The latter two, k-means and GMM, only consider the shape of the crowd blobs and ignore the density values.

Because the original-resolution density maps from CNN-pixel have well-defined peaks similar to the ground-truth density maps (see Fig. 2h), we also propose a modification of GMM, where each pixel in the crowd segment is weighted according to its density value. This weighting process makes the GMM cluster centers move towards the high density value region, which is more likely to be the real location of the object. In other words, this weighting operation better tries to find local peaks. In practice, we implement the weighting by discretizing the density map, and then repeating pixel locations according to their discretized density values, where locations with high density values appear more than once. GMM clustering is then applied to the samples.

D. Improving Tracking with Density Maps

We next describe how density maps can be used to improve state-of-the-art tracking of people in crowds. Recently, kernelized correlation filters (KCF) has been widely used for single object tracking [37, 38]. However, tracking a person in a crowd is still challenging for KCF due to occlusion and background clutter, as well as low-resolution targets. Here we combine the KCF tracker with the crowd density map so as to focus the tracker on image regions with high object density, which are more likely to contain the target. This effectively prevents the tracker from drifting to the background. In particular, our fusion strategy is to element-wise multiply the KCF response map with the corresponding crowd density map. The maximum response of the combined map is then selected as the tracked position. An example is shown in Fig. 4.

IV. Density Map Quality

As high-quality per-pixel reproductions of the density map will likely yield better localization (detection and tracking) performance, in this section, we measure the quality of high-resolution density maps using various attributes. We then relate these measures to localization performance in the next section.

We compare the following density map methods: MESA [3], ridge regression (RR) [4], density-patch CNN (CNN-patch) [8], multi-column CNN (MCNN) [10], as well as full-resolution density maps produced by our pixel-wise CNN (CNN-pixel) and fully-convolutional NN (FCNN-skip). We also test MCNN followed by upsampling-convolution layers to get full resolution density maps, denoted as MCNN-up. All the predicted density maps are from UCSD "max" dataset (see Section V-B) and implementation details are in Section V-A.

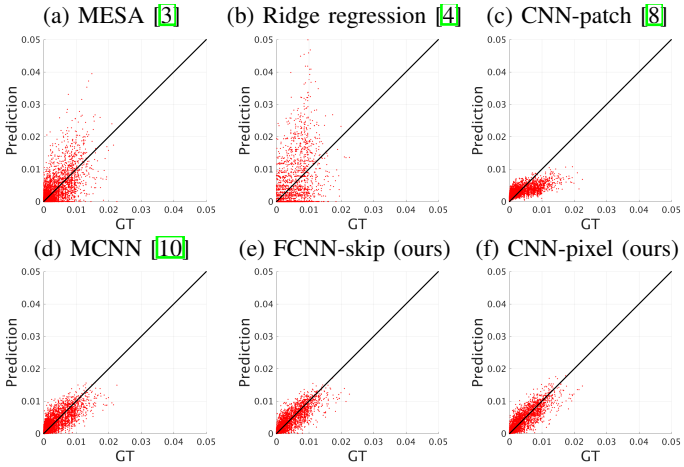


Figure 5: Scatter plots of GT density and predicted density values.

A. Per-pixel Reproduction

We first consider how well each method can reproduce the per-pixel values in the ground-truth density map. Fig. 5 shows a scatter plot between the ground-truth density pixel values and the predicted density pixel values for the various methods. MCNN, CNN-pixel and FCNN-skip show the best correlation between ground-truth and prediction, and are more concentrated around the diagonal. Ridge regression (RR) tends to over-predict the density values, because the filtering of negative feature weights compacts the density map causing higher density values. On the other hand, CNN-patch [8] under-predicts the values, because the low-resolution predictions and patch-wise averaging spreads out the density map causing lower overall density values. MESA [3] predictions are also concentrated around the diagonal, but less so than MCNN, FCNN-skip, and CNN-pixel. This is because MESA optimizes the count prediction error within rectangular sub-regions of the density map, rather than per-pixel error.

B. Compactness and Localization

Compactness and localization are two important properties for using density maps for detection and tracking. Compactness means the density values are concentrated in tight regions around a particular position, while localization means that these positions are located near a ground-truth dot annotation.

To measure the compactness and localization, we place a bounding box (scaled for perspective) over each dot annotation (ground-truth position). If a density map is not compact, then some density will leak outside the bounding boxes. The amount of leakage can be measured by calculating the ratio of density inside the bounding boxes to the total density in the image, which is denoted as *bounding box density ratio* (BBDR). If the prediction is localized well, then the total predicted density inside the bounding boxes should match the

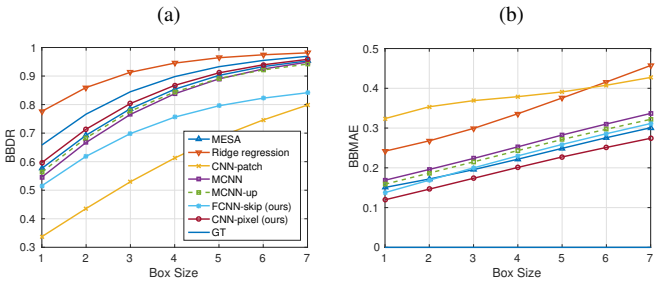


Figure 6: Measures of compactness and localization: (a) bounding box density ratio (BBDR). Higher values means more compact, e.g. ridge regression prediction is the most compact. (b) bounding box MAE (BBMAE). Lower values mean better localization.

corresponding total ground-truth density. Hence, the measure of localization is the MAE inside all boxes, denoted as *bounding box MAE* (BBMAE).

Figs. 6a and 6b plot the curves for BBDR and BBMAE by varying the size of the bounding box. CNN-pixel has the best localization measure (lowest BBMAE), and also has similar compactness (BBDR) to the ground-truth. FCNN-skip also has good localization (low BBMAE), but, in contrast, its density maps are more spread out (low BBDR, less compact) due to its upsampling operation. MCNN-up has slightly higher BBDR and lower BBMAE compared with MCNN, which suggests that the learned upsampling layer can improve compactness and localization, over bicubic upsampling. RR has the most compact density maps (BBDR even higher than the ground-truth density used for training), but has poor localization (high BBMAE) because the centroid of the local modes are shifted.

C. Temporal Smoothness

We measure temporal smoothness of the density maps and the detection results. Density maps and detections that are smooth in time suggest that finding each person's trajectory should be easier. Hence, these temporally smooth density maps could aid object tracking. Table II shows the mean absolute difference (MAD) between density maps of consecutive frames. The temporal smoothness of deep learning methods are better than traditional methods even though they do not directly use frame difference information. This shows that the CNN feature extractor produces smoothly varying feature representations, compared to the random forest representation used with MESA/RR. The density maps of CNN-patch are the most temporally smooth (lowest MAD), but this is because those maps are more spread out (less compact), which cannot help to localize and track the object (refer to Section V-D).

To measure the temporal smoothness of the detections, we calculate the error vector between a detection location and its corresponding ground-truth. The error distance (ED) measures the distance to the ground-truth, and is equal to the length of the error vector. We also measure the length of the difference between two consecutive error vectors corresponding to the same ground-truth trajectory, and denote it as EDD (error difference distance). EDD measures the stability of the detected point over time, when compared to the GT trajectory. Finally, we measure the miss rate, which is the percentage of missed detections. Table II shows results for IntProg and GMM-weighted. With IntProg, CNN-pixel and FCNN-skip have similar ED, and CNN-pixel has the lowest EDD, which indicates that the detected points are more stable over time for CNN-pixel. Using GMM-weighted, CNN-pixel and FCNN-skip obtain the most accurate detected positions (lowest ED), but CNN-pixel is more temporally stable. These results suggest that temporally smooth density maps, produced using per-pixel prediction, could be helpful for object tracking, e.g., by incorporating them into a tracking-by-detection framework.

D. Summary

In summary, CNN-pixel performs best on various metrics, including compactness (BBDR), localization (BBMAE), and temporal smoothness (EDD). In contrast, reduced-resolution maps that require either fixed upsampling (CNN-patch and MCNN) or learned upsampling-convolution (FCNN-skip and MCNN-up) had worse quality metrics. The downsampling operations in the CNN obfuscate the true position of the people in the reduced-resolution maps. This loss of spatial information can only be partially compensated using learned upsampling and skip connections.

V. Experiments

In this section we present experiments on several crowd analysis tasks to compare density maps estimated by recent methods, including MESA [3], ridge regression (RR) [4], Regression forest [7], COUNT forest [9], density-patch CNN (denoted as CNN-patch) [8], multi-column CNN (MCNN) [10], Hydra CNN [14] and CNN-boost [24], as well as full-resolution density maps produced by our CNN-pixel and FCNN-skip. We first present experiments on crowd counting using benchmark datasets for people and cars. We then present experiments on people detection using density maps, to compare the localization ability of the density maps. Next we conduct experiments on people tracking by fusing the density maps and visual tracker response, in order to compare the localization ability and the temporal smoothness of the density maps. Finally, we perform an ablation study on CNN-pixel.

A. Implementation Details

The CNN-pixel architecture appears in Fig. 3. The input image patch is a 33×33 grayscale image. Every convolution layer is followed by an across channel local response normalization layer and a max pooling layer. The first convolutional layer contains 64 5×5×1 filters (i.e., 5×5 filter on 1 grayscale channel), and the max-pooling is 2×2 with a stride size of 2×2. The second convolutional layer contains 64 5×5×64 filters, and the max-pooling is 3×3 pooling with a stride size of 2×2. When the images are high resolution and contain large objects, as in the TRANCOS dataset [39] (640×480) and WorldExpo'10 dataset [8] (720×576), we use a 65×65 image patch and add one additional convolution-pooling layer to capture enough context information, and to keep the final output feature map a consistent size. The CNN-pixel is trained using the regression and classification tasks, as described in Section III-A. The training set is constructed by densely extracting all gray-scale patches, whose centers are inside the ROI, from the training images. The network is trained with standard back-propagation [40]. Momentum and weight decay

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2837153, IEEE Transactions on Circuits and Systems for Video Technology

		IntProg			GMM-weighted		
Density maps	MAD	ED	EDD	Miss rate	ED	EDD	Miss rate
MESA [3]	16.34	4.96±0.71	4.89±0.92	13.96%	2.90±0.74	2.63±0.81	8.69%
Ridge regression [4]	18.34	4.48±0.69	4.45±0.95	15.87%	2.91±0.81	2.33±0.76	13.14%
CNN-patch [8]	**1.89**	5.40±0.74	4.23±1.17	41.03%	4.84±0.91	4.50±1.45	16.66%
MCNN [10]	3.96	4.22±0.76	3.32±0.90	19.77%	3.56±0.76	2.24±0.79	10.53%
MCNN-up	3.96	4.13±0.89	3.12±1.03	16.13%	3.52±0.84	2.03±0.70	10.54%
CNN-pixel (ours)	3.96	**3.61**±0.72	**2.90**±0.83	9.82%	2.78±0.85	**1.88**±0.75	**8.43%**
FCNN-skip (ours)	4.78	**3.61**±0.74	3.38±1.01	13.34%	**2.73**±0.76	2.26±0.84	10.21%
Ground truth	3.31	3.06±0.68	2.30±1.22	2.59%	1.29±1.01	1.18±0.93	5.24%

Table II: Comparisons of temporal smoothness of density maps and detections.

are used, and one dropout layer is used after every non-output fully connected layer (refer to Fig. 3) to reduce over-fitting.

Our FCNN-skip is adapted from the CNN-pixel model. The structure of the lower convolutional layers and pooling layers remains the same as CNN-pixel. As in [25], the first fully connected layer is adapted to a convolutional layer whose filter size is the same as the output feature map size produced by our CNN-pixel model (9×9). The remaining fully connected layers are implemented as convolutional layers using a filter size of 1×1. For the skip-branches, a convolutional layer with 1×1×64 filters is applied to the lower convolutional layer, and then merged with the density map using an element-wise sum.

The training of the FCNN-skip network is initialized with the weights from CNN-pixel network. The weights of the convolutional layer after up-sampling are initialized as an average filter. The weights of the skip branches are initialized as zeros (as in [25]), so that the training first focuses on tuning the initial weights (from CNN-pixel) to the FCNN architecture.

In the inference stage, CNN-pixel densely extracts test patches using the training patch size, and predicts the density values in a pixel-wise manner. FCNN-skip takes any size image as input and predicts the whole density map. Both CNN-pixel and FCNN-skip density maps do not have block artifacts caused by combining overlapping patch predictions. The count in the region-of-interest of an image is the sum of the predicted density map inside the region. We do not apply any post-processing to the estimated count, cf., the ridge regression used in [8] to remove systematic count errors. Finally, note that we do not change the input patch size according to its location in the scene, which is a form of perspective normalization.

B. Counting Experiments

We first compare the density estmation methods on three crowd datasets, UCSD pedestrian, WorldExpo'10, and UCF_CC_50, and a car dataset, TRANCOS. The statistics of the datasets are summarized in Table III, and example images can be found in the supplementary.

Dataset	N_f	Res.	Range	T_p
UCSD [1]	2000	238×158	11-45	49885
WorldExpo'10 [8]	3980	720×576	1-253	199923
UCF_CC_50 [5]	50	varies	96-4633	63974
TRANCOS [39]	1244	640×480	9-107	46796

Table III: Statistics of the four tested datasets. N_f is the number of annotated images or frames; Res. is the image/frame resolution; Range is the range of number of objects inside the ROI of a frame; T_p is the total number of labeled objects.

1) Datasets: The UCSD dataset is a low resolution (238×158) surveillance video with perspective change and heavy occlusion between objects. It contains 2000 video frames and 49,885 annotated pedestrians. On UCSD, we test the performance on the traditional setting [1] and four down-sampled splits [2]. In the first protocol [1], all frames from 601-1400 are used for model training and the remaining 1200 frames are for testing. The second protocol is based on four downsampled training splits [2]: "maximal" (160 frames, 601:5:1400), "downscale" (80 frames, 1201:5:1600), "upscale" (60 frames, 801:5:1100), and "minimal" (10 frames, 641:80:1361). These splits test robustness to the amount of training data ("maximal" vs. "minimal"), and generalization across crowd levels (e.g., "upscale" trains on small crowds, and tests on large crowds). The corresponding test sets cover all the frames outside the training range (e.g. 1:600 and 1401:2000 for "maximal"). The ground-truth density maps use $\sigma = 4$.

The WorldExpo'10 dataset [8] is a large scale people counting dataset captured from the Shanghai 2010 WorldExpo. It contains 3,980 annotated images and 199,923 annotated pedestrians from 108 different scenes. Following [8], we use a human-like density template, which changes with perspective, to generate the ground-truth density map, and the fractional density value in the ROI is used as the ground-truth since there are many people near to the ROI boundary. Models are trained on the 103 training scenes and tested on 5 novel test scenes.

The UCF dataset [5] contains 50 very different images with the number of people varying from 96 to 4,633. In contrast to the previous two datasets, UCF only contains one image for each scene, and each image has a different resolution and camera angle. This dataset measures the accuracy, scalability and practicality of counting methods, as well as tests how well methods handle extreme situations. The ground-truth density map is generated using $\sigma = 6$. Following [5, 8], the evaluation is based on 5-fold cross-validation.

The TRANCOS dataset [39] is a benchmark for vehicle counting in traffic congestion situations, where partial-occlusion frequently occurs. It has training, validation and test sets, containing 1,244 annotated images from different scenes and 46,796 annotated vehicles. Following the first training strategy in [39], we use the training and validation data as the training and validation sets for model training, and evaluate the model on the test set. Different from the WorldExpo and UCF datasets, TRANCOS dataset contains multiple scenes but the same scenes appear in the training, validation, and test sets. The ground-truth density map is generated using $\sigma = 10$.

Counting predictions are evaluated using the mean absolute error (MAE) or mean square error (MSE) with respect to the ground truth number of people. The fractional number is used for WorldExpo and integer number is used for the other

Method	MAE	MSE
Kernel Ridge Regression [41]	2.16	7.45
Ridge Regression [22]	2.25	7.82
Gaussian Process Regression [1]	2.24	7.97
Cumulative Attribute Regression [21]	2.07	6.86
COUNT forest [9]	1.61	4.4
CNN-patch+RR [8]	1.60	3.31
MCNN [10]	**1.07**	**1.35**
CNN-boost fine-tuned using 1 boost [14]	1.10	-
CNN-pixel (ours)	1.12	2.06
FCNN-skip (ours)	1.22	2.25

Table IV: Test errors on the UCSD dataset when using the whole training set.

datasets, following the convention.

2) UCSD pedestrian dataset: We present results on the two experimental protocols for UCSD. The results for the first protocol (full training set) are shown in Table IV, where the last five CNN methods are based on density maps, and the remaining are traditional regression-based methods. The CNN-based density estimation methods have lower error than the traditional methods, especially the last four methods which reduce error by a large margin. MCNN has slightly lower MAE than CNN-boost and CNN-pixel (1.07 vs 1.10 and 1.12). FCNN-skip method achieves worse performance than CNN-pixel (1.22 vs 1.12) but consumes much less time in the prediction stage (16 ms per frame vs. 4.6 sec per frame on UCSD). In comparison, MCNN takes 31 ms and CNN-patch takes 37 ms per frame respectively.

Method	Max	Down	Up	Min	Avg
MESA [3]	1.70	1.28	1.59	2.02	1.65
Regression forest [7]	1.70	2.16	1.61	2.20	1.92
Ridge regression [4]	**1.24**	1.31	1.69	**1.49**	1.43
COUNT forest [9]	1.43	1.30	1.59	1.62	1.49
CNN-patch+RR [8]	1.70	**1.26**	1.59	1.52	1.52
MCNN [10]	1.32	1.71	2.05	1.56	1.66
CCNN [14]	1.65	1.79	**1.11**	1.50	1.51
Hydra 2s [14]	2.22	1.93	1.37	2.38	1.98
CNN-pixel (ours)	1.26	1.35	1.59	**1.49**	**1.42**
FCNN-skip (ours)	**1.24**	1.38	2.13	1.83	1.65
CNN-pixel-VS	1.48	—	—	—	—

Table V: Comparison of MAE between density-based counting methods on the UCSD dataset using 4 training splits.

Table V shows the MAE for the four training splits of the second protocol. Here we only list results of density-based approaches, which represent the current state-of-the-art counting performance. Over the four splits, CNN-pixel and RR have the lowest average MAE. MCNN and FCNN-skip, which are both FCNNs, performs a little worse, mainly on the "up" split, than CNN-patch and CNN-pixel but are more computationally efficient. Examining the error heat maps in the supplementary shows that the FCNN-skip errors are overall larger on the "up" split, and without any systematic bias.

One advantage of CNN-based method is that they can improve their performance with more training data. When using the full training set (see Table IV for methods trained with the full training set), both CNN-pixel and MCNN can lower their MAE compared with the "max" split in Table V, which uses only 1/5 of the training set. An example density map estimated by our CNN-pixel model is shown in Fig. 7, and Fig. 2 compares the density maps from different methods.

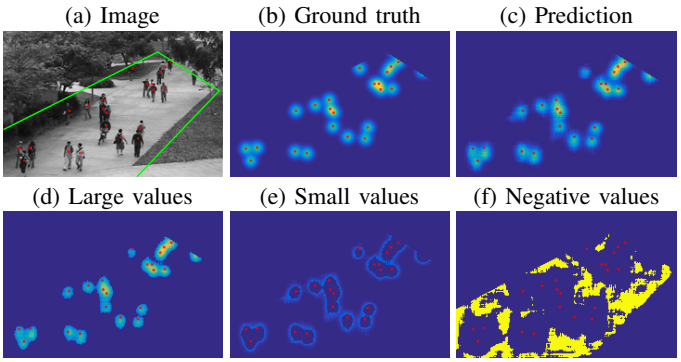


Figure 7: An example density map produced by our CNN-pixel model. Our density maps are concentrated around the ground-truth annotations with little "leakage" of density into the background. The sum of the negative values is very small (-0.10), and does not affect the counting, detection or tracking performance.

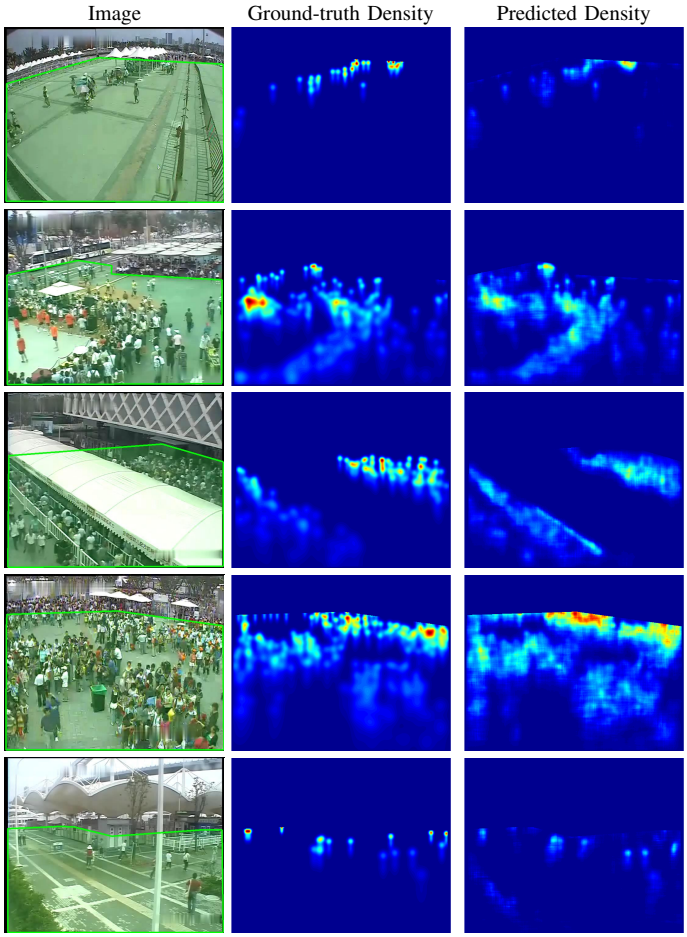


Figure 8: Results on WorldExpo dataset using our CNN-pixel model.

3) WorldExpo'10 dataset: The results on the WorldExpo'10 dataset [8] are shown in Table VI and Fig. 8. MCNN has the lowest average error, while CNN-pixel, FCNN-skip and CNN-patch without fine-tuning perform similarly. Looking at the individual scenes, different methods perform better on each scene. All the methods have larger errors on Scenes 2, 3 and 4, compared with Scenes 1 and 5. These three scenes contain more people on average, while only 12% of all the training frames contain large crowds (>80 people).

Method	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Average
LBP+RR	13.6	58.9	37.1	21.8	23.4	31.0
CNN-patch [8]	10.0	15.4	15.3	25.6	4.1	14.1
CNN-patch + cross-scene fine-tuning [8]	9.8	**14.1**	14.3	22.2	**3.7**	12.9
MCNN [10]	3.4	20.6	**12.9**	**13.0**	8.1	**11.6**
CNN-pixel (ours)	**2.9**	18.6	14.1	24.6	6.9	13.4
FCNN-skip (ours)	3.9	16.9	19.3	27.7	5.6	14.7

Table VI: Mean absolute error (MAE) on WorldExpo'10 dataset.

Method	MAE	Std
MESA [3]	493.4	487.1
Density-aware [12]	655.7	697.8
FHSc [5]	468.0	590.3
FHSc + MRF [5]	419.5	541.6
CNN-patch [8]	467.0	498.5
MCNN [10]	377.6	509.1
Hydra 2s [14]	**333.7**	425.3
CNN-boost fine-tuned using 1 boost [14]	364.4	**341.4**
CNN-pixel (ours)	406.2	404.0
FCNN-skip (ours)	431.6	379.6

Table VII: Mean absolute error (MAE) on the UCF dataset. Std is the standard deviation of the MAE.

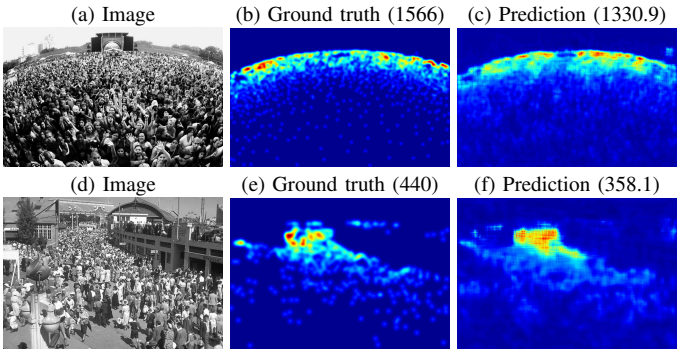


Figure 9: Example results on UCF_CC_50 dataset using our CNN-pixel model. The number in parenthesis is the count.

On Scene 3, CNN-pixel under-predicts the density because there are many people in the upper-right that are partially-occluded by the roof, leaving just the heads visible. Because the roof is in the ROI, the ground-truth density has non-zero values on the roof (see Fig. 8). However, CNN-pixel predicts zero density on the roof region (because it sees no human parts there), which causes the count to be under-predicted. For Scene 4, the ROI boundary cuts through a large crowd in the background. A large portion of the density is outside the ROI, due to the human-shaped ground-truth, where half of the density is contributed from the small head region. As a result CNN-pixel predictions tend to be larger due to this confounding effect (refer to the count plot and the error heat maps in the supplementary.).

4) UCF_CC_50 dataset: The results on the UCF dataset are presented in Table VII, and examples of our predicted density maps appear in Fig. 9. The CNN-based methods show their capability of handling these extremely crowded images. Note that although there are only 40 training images, there are still many patches of people that can be extracted to train the CNN from scratch.

5) TRANCOS dataset: For TRANSCOS, the evaluation metric is the Grid Average Mean absolute Error (GAME) [39],

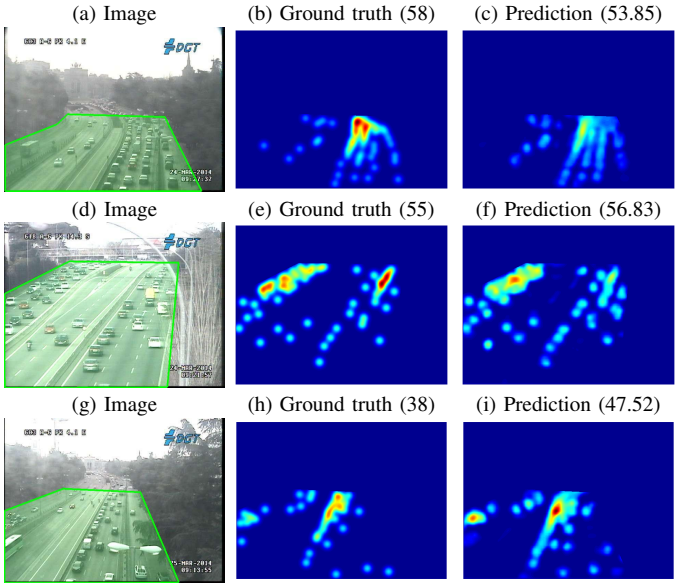


Figure 10: Example results on TRANCOS using our FCNN-skip model. The number in parenthesis is the count.

which measures the error within grid sub-regions,

$$\text{GAME}(L) = \frac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{4^L}|\hat{c}_n^l - c_n^l| \tag{7}$$

where, $\hat{c}_n^l$ and c_n^l are the estimated count and ground truth count in region l of image $\mathcal{I}_n$. The number of sub-regions is determined by the level L. For $L = 0$, GAME(0) is equivalent to MAE. The results are shown in Table VIII and example predictions are presented in Fig. 10. The CNN methods surpass the methods using traditional features by a large margin, especially the last three in Table VIII. FCNN-skip has the lowest MAE, GAME(1), and GAME(2) among the methods.

C. Detection Experiments

In this subsection, we test the performance of density maps for people detection.

1) Setup: We use the integer programming method proposed in [11] (IntProg), as well as 3 baselines described in Section III-C: finding local peaks (Local-max), K-means clustering, and GMM clustering. We also test our proposed weighted GMM clustering (denoted as GMM-weighted).

For comparison, we run the detection algorithms on a variety of predicted density maps, as well as the ground-truth density map, which provides a reference for detection using these density maps. We perform detection on density maps of the same resolution of its input image. We use the original resolution density maps from CNN-pixel, FCNN-skip, MESA [3], and RR [4]. Since the predicted density maps from MESA and RR are noisy, a Gaussian kernel is used to smooth the density maps (std. 3 for LocalMax and

Method	GAME(0), MAE	GAME(1)	GAME(2)	GAME(3)
Regression forest [7] + RGB Norm + Filters	17.68	19.97	23.54	25.84
MESA [3] + SIFT	13.76	16.72	20.72	24.36
HOG-2 [42]	13.29	18.05	23.65	28.41
CNN-patch [8]	11.24	12.36	14.51	18.67
Hydra 3s [14]	10.99	13.75	16.69	19.32
MCNN [10]	7.51	9.12	11.50	**15.85**
CNN-pixel (ours)	5.87	8.63	11.43	16.31
FCNN-skip (ours)	**4.61**	**8.39**	**11.08**	16.10

Table VIII: Evaluation on the TRANCOS car dataset.

std. 2 for all the other detection methods). For CNN-patch [8] and MCNN [10], we upsample the reduced-resolution density maps to the original resolution using bicubic interpolation. As MCNN shows strong performance in the counting task, but only produces reduced resolution density maps, we also test a variant of MCNN that predicts a full-resolution density maps (denoted as MCNN-up). Similar to FCNN-skip, two learned upsampling-convolution layers, each with a $3 \times 3 \times 1$ filter, are used to upsample the MCNN predicted density map. We follow the same evaluation procedure as [7, 11]. Detections are *uniquely* paired with ground-truth locations within a matching distance, and precision, recall, and F1 score are calculated.

2) Results: Table IX presents the detection results on UCSD using the various density maps and detection methods. Using IntProg with density maps from CNN-pixel or MESA yields the best F1 score. Although the density maps from MESA look "noisy" (e.g., see Fig. 2c), actually the location information is preserved well by the MESA criteria, as its BBMAE is 2nd lowest among the methods (see Fig. 6b). However, MESA density maps are less compact than most other CNN-based methods (lower BBDR in Fig. 6a), and cannot maintain the monotonicity (per-pixel reproduction Fig. 5), and thus detection with GMM-weighted is worse than MCNN and MCNN-up. Detection with CNN-pixel has higher precision than MESA, due to the more compact density maps of CNN-pixel, but also lower recall.

The detection result for FCNN-skip is worse than CNN-pixel because its density maps are less compact (lower BBDR in Fig. 6a) and less isolated due to the upsampling process. While the upsampling and skip connections compensate for the loss of spatial information, there is still a degradation in detection accuracy.

Using the upsampling-convolution layers with MCNN (MCNN-up) improves the F1 score on all methods compared to MCNN, although the counting performance decreases slightly (1.37 vs. 1.32). The improved detection performance of MCNN-up is due to better compactness (higher BBDR) and localization (lower BBMAE) of its density maps, compared to MCNN (see Fig. 6). Because the objects in UCSD can be smaller than 20 pixels, a downsample factor of 4 has a large influence on localization – a full resolution density map is crucial for good localization performance, as compared to bicubic upsampling of the reduced-resolution density map.

Detection accuracy on RR maps is not as high as those by MESA or CNN-pixel with IntProg and GMM-weighted. To maintain positive density values, RR uses a heuristic to remove feature dimensions with negative weight during the training process. Most of these situations affect the boundary regions of the density blobs (see Fig. 2d), resulting in very compact density maps (highest BBDR in Fig. 6a). Yet, the disadvantage of this feature removal is that it shifts the center of mass of the density blobs, resulting in worse localization metrics (lower BBMAE, see Fig. 6b), which affects the detection accuracy with IntProg and GMM-weighted. On the other hand, the density blobs are very compact, which favors the K-means and GMM detection methods, which only consider the shape of the density blobs.

Although CNN-patch is suitable for crowd counting, the detection results are worse than other density maps. CNN-patch generates a reduced-resolution density maps and also averages overlapping density patch predictions, which smooths out local density peaks (lowest BBDR in Fig. 6a), thus making localization of each person more difficult.

We note that two attributes are important for good detection performance using the tested detection methods: 1) accurate peaks with good monotonicity, otherwise the detections easily drifts; 2) having more compact (higher BBDR) and isolated peaks reduces the search space and makes detection of every individual object easier. Using higher quality density maps (such as from CNN-pixel), the simple GMM-weighted method, which considers both the shape and density of the blobs, achieves very similar performance to the more complex IntProg. Finally, there exists a large gap between detection on the ground-truth density map and the predicted density maps (e.g., using IntProg, F1 of 97.96 vs 92.89). Hence, there still is room to improve density map methods such that the location information is better preserved.

D. Tracking Experiments

We next test the ability of density maps to improve pedestrian tracking algorithms.

1) Setup: We test the fusion method described in Section III-D to combine the response map of the kernel correlation filter (KCF) with the crowd density map. The position in the resulting map with the maximum response is used as the tracked location, which is then passed to the tracker for online updating the KCF appearance model. We test KCF fusion with the same density maps used for the detection experiment (all trained on UCSD "max" split), as well as the original KCF without fusion. The performance of single person tracking is evaluated on each person in the testing set of the UCSD dataset (1200 frames). In particular, we plot a precision-threshold curve, which shows the percentage of frames where the tracking error is within a distance threshold. The tracker is initialized with the ground truth location of a person after they have fully entered the video.

2) Results: Fig. 11 shows the tracking performance. Fusing the density map with the KCF response map can improve

	IntProg			GMM-weighted			GMM			K-means			Local-max		
Density map	R%	P%	F1%	R%	P%	F1%	R%	P%	F1%	R%	P%	F1%	R%	P%	F1%
MESA [3]	92.39	91.18	91.78	91.31	86.69	88.94	81.55	78.78	80.14	85.34	81.75	83.50	81.01	86.58	**83.70**
Ridge regression [4]	85.04	89.26	87.10	86.86	84.27	85.55	86.80	80.59	**83.58**	87.74	81.46	**84.49**	72.90	88.00	79.74
CNN-patch [8]	59.28	85.82	70.12	83.34	78.45	80.82	51.57	49.28	50.40	55.99	53.49	54.71	39.55	83.64	53.70
MCNN [10]	80.27	93.90	86.55	89.47	89.78	89.63	79.36	78.82	79.09	81.82	81.26	81.54	70.44	85.46	77.22
MCNN-up	83.87	94.87	89.03	89.46	90.84	90.14	81.20	81.68	81.44	83.70	84.19	83.95	72.49	83.34	77.54
FCNN-skip (ours)	88.29	88.19	88.24	91.77	86.20	88.90	82.44	76.92	79.59	85.40	79.69	82.45	74.77	84.54	79.35
CNN-pixel (ours)	90.19	95.75	**92.89**	91.57	89.82	**90.69**	80.05	78.38	79.20	82.68	80.95	81.81	71.19	87.12	78.36
CNN-pixel-VS	79.26	94.64	86.27	87.40	92.03	89.66	75.04	79.61	77.26	77.54	82.26	79.83	62.29	91.45	74.11
CNN-pixel (ours) [full]	88.02	97.45	92.49	89.90	92.46	91.16	78.55	80.84	79.68	80.80	83.15	81.96	73.11	86.78	79.36
GT density map	97.41	98.51	97.96	94.76	94.56	94.66	85.25	86.64	85.94	87.78	89.05	88.41	80.51	84.88	82.64

Table IX: Detection performance on UCSD dataset ("max" split) using density maps with different detection methods. For comparison, SIFT-SVM [3] obtains an F1 score of 68.46 and region-SVM [43] obtains an F1 score of 89.53. [full] denotes training with the full 800 frames of the training set. Bold numbers indicate best performance for each detection method among the density maps.

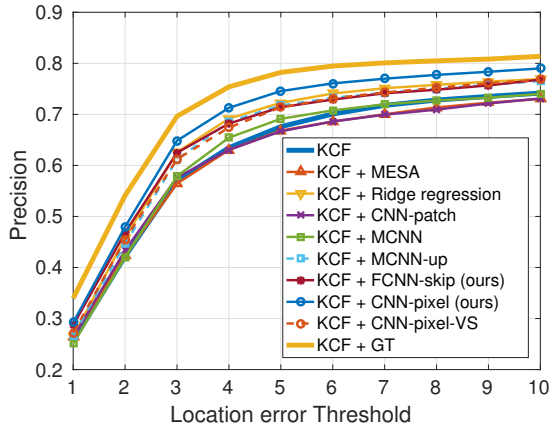


Figure 11: Single object tracking performance on UCSD.

tracking performance. Specifically, using CNN-pixel yields the largest increase in precision compared to other methods (e.g., P@4 of 0.713 for CNN-pixel vs. 0.692 for RR), due to its density maps having better localization metric (lower BBMAE, see Fig. 6b) and its detections are temporally most stable (low ED and EDD in Table II). Fusion with the CNN-patch density maps does not improve the tracker because those density maps are too smooth, and thus the fused map does not change much. Fusion with MCNN-up improves the tracking performance over MCNN, because MCNN-up density maps are more compact and better localized (see Fig. 6), resulting in temporally more stable detections (see Table II). Again, this shows that original resolution density maps are preferred for localization tasks. Finally, fusing the ground-truth density map with KCF yields the best tracking results, which marks the upper bound of the performance for this type of fusion.

E. Summary

In summary, in the experiments, we have evaluated density maps for counting, detection, and tracking. For counting, the resolution of the density map does not affect much the counting accuracy. Methods that produce reduced-resolution maps, e.g., MCNN, can perform well. In contrast, using original-resolution density maps gives better accuracy for localization tasks, such as detection and tracking. In particular, per-pixel generation of original-resolution density maps without using upsampling (e.g., CNN-pixel) have higher fidelity (better compactness and localization metrics, BBDR and BBMAE) than those that are upsampled from reduced resolution maps (e.g., FCNN-skip, MCNN, MCNN-up), leading to better detection and tracking performance. For FCNN-skip, MCNN, and MCNN-up, downsampling in the CNN obfuscate the true position of the people in the reduced-resolution maps. This loss of spatial information can only be partially compensated using learned upsampling and skip connections.

F. Training and Architecture Variations

Next we report results of using various common strategies for training and alternative architectures for CNN-pixel and FCNN-skip. Experiments were run on the UCSD dataset using the "max" split, and the results are summarized in Table X.

Method	Variation	MAE
CNN-pixel	–	1.26
CNN-pixel	varying std. with perspective	1.48
CNN-pixel	fine-tuning with only regression task	1.20
CNN-pixel	feature combo	1.21
CNN-pixel	256 neurons for the first FC	1.38
CNN-pixel	4 or 6 convolution layers	1.26, 1.38
CNN-pixel	6 or 12 residual blocks [29]	1.26, 1.34
CNN-pixel	2 dense blocks (2, 4 or 8 layers each) [44]	2.77, 1.82, 2.64
FCNN-skip	–	1.24
FCNN-skip	only pixel-wise loss	1.41
FCNN	only pixel-wise loss	1.54
FCNN	only count loss	1.82
FCNN	hole convolution [35, 45]	1.93

Table X: Comparison of variations in training and architectures of CNN-pixel and FCNN-skip. All the experiments are on the UCSD dataset using the "max" split.

Data augmentation: Because we extract all the possible image patches to train the network, no random translations are needed when cropping patches. We also added small random Gaussian noise and horizontally flipped the patches, but no obvious improvement was observed. Most likely this is because the dense patch sampling already covers many useful permutations of the input.

CNN-pixel fine-tuning with only the regression task: As the classification task is an auxiliary task for guiding the CNN during training, it is possible that it could adversely affect the training of the density regressor. Starting from a well-trained network, we removed the classification task and fine-tuned the network only using the regression task. However, with fine-tuning we obtained similar regression performance (MAE: 1.20). Most likely this is because counting (implemented as classification) and density prediction are related tasks, which can share common image features, so it is not necessary to explicitly fine-tune the density regressor alone.

CNN-pixel with combined features from different convolution layers: A commonly used strategy to improve the performance of CNNs is to combine features from different convolution layers [30, 31]. We use a fully connected layer (in contrast to the convolutional layer used in the FCNN-skip adaptations) to collect features from different convolutional layer and concatenate them together. Similar performance to the original CNN-pixel was observed (MAE: 1.21).

CNN-pixel with more convolution layers: Recently, very deep networks have been used for high-level classification and semantic segmentation tasks [26–29]. However, we note that density estimation task is a mid-level task, and thus does not necessarily require very deep networks. We test CNN-pixel variations with 4 or 6 convolution layers, with 2 pooling layers. No gain is observed using 4 convolution layers (MAE: 1.26), and performance decreases slightly (MAE: 1.38) when using 6 convolution layers. We also tested a ResNet version, which replaces the convolution layers with residual blocks, and did not see better performance (MAE: 1.26). We also tried several versions of DenseNet [44] with 2, 4, or 8 dense layers in each dense block. The more complicated network has poor performance on the counting task, possibly due to the limited training data relative to the network size. A similar trend is also observed in [24].

CNN-pixel with smaller fully-connected layer: CNN-pixel uses 512 neurons in the first fully-connected (FC) layer which seems large for such a shallow network. We test a version of CNN-pixel with fewer neurons in the first FC layer. Using 256 neurons gives worse performance (MAE: 1.38), possibly because more neurons are needed to capture all the appearance variations associated with the same density value.

FCNN-skip with only pixel-wise loss: Our FCNN-skip is trained with both the patch-wise count loss and a pixel-wise loss. Removing the count loss from the training leads to higher error (MAE: 1.41). This shows the benefit of including the count loss, which focuses on reducing systematic counting errors, which cannot be well reflected in the pixel-wise loss.

FCNN without spatial compensation: We also compared FCNN-skip with an FCNN without skip branches (denoted as FCNN). Here we only train with the pixel-wise loss, and the error increases when no skip branches are used.

FCNN with only count loss: Next, we train FCNN with only the count loss, which results in the predicted density map spreading out, making the high density and low density regions more similar (see Fig. 12). This leads to poor counting performance (MAE: 1.82), and hence the pixel-wise loss is required to maintain the structure of the density map.

Effect of Multi-task learning: We compared the effect of different λ_2 on multi-task learning. For λ_2 ={0, 0.1, 1, 10, 100}, CNN-pixel gets MAE of {1.41, 1.27, 1.26, 1.44, 1.47}. Using $\lambda_2 = 0.1$ gives similar performance to $\lambda_2 = 1$, while other settings decrease the performance. Similar observations are also reported in [30].

Effect of varying Gaussian widths: We have conducted experiments to understand the effect of including perspective in the ground-truth densities for UCSD. As in [8], the standard deviation of the Gaussian is varied using the perspective map. Results using CNN-pixel with the new density maps (denoted as "CNN-pixel-VS") for counting, detection, and tracking are shown in Tables V and IX, and Fig. 11. The counting results using CNN-pixel-VS are worse than CNN-pixel (MAE 1.48 vs 1.24). Detection performance also is worse, since IntProg and GMM-weight use the predicted count as either a constraint or the number of clusters. For tracking, CNN-pixel-VS also performs worse than CNN-pixel, mainly because the spread out density maps have lower values, which decreases the influence of the density map on the tracking response map.

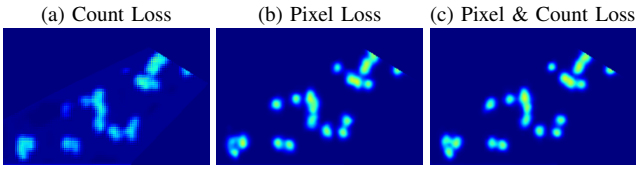


Figure 12: Example of using different loss functions to train FCNN.

FCNN using hole convolution: Another method to get a full resolution density map is to use "hole convolution" [35, 45], which changes all the stride sizes to one and adds zeros into the filters of the convolutional and pooling layers. We first adapt our CNN-pixel model to FCNN. After the parameters have been trained, we use the hole algorithm to predict a density map with the same resolution as the input. This approach has poor performance (MAE: 1.93) while taking more time than FCNN with up-sampling because it need to compute responses for every image pixel. In contrast, using FCNN with up-sampling is more suitable since it encourages smooth density maps.

VI. Conclusion

In this work, we compare crowd density maps, produced by different methods, on several crowd analysis tasks, including counting, detection, and tracking. While reduced-resolution density maps produced by fully-convolutional NN (e.g., MCNN) perform well at counting, their accuracy diminished at localization tasks due to the loss of spatial resolution, which could not be completely recovered using upsampling and skip connections. In contrast, dense pixel-prediction of a full resolution density map, using CNN-pixel, produced the highest quality density map for localization tasks, with slight degradation for the counting task. However, dense prediction suffers from higher computational complexity, compared to fully-convolutional networks.

We also proposed several metrics for measuring aspects of the density maps to explain why those density maps with similar counting accuracy can perform differently on detection and tracking. These metrics can help to guide future research in designing density map methods to estimate high quality density maps for both counting and localization tasks.

References

[1] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[2] D. Ryan, S. Denman, C. Fookes, and S. Sridharan, "Crowd counting using multiple local features," in *Digital Image Computing: Techniques and Applications*, 2009.

[3] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Advances in neural information processing systems*, 2010.

[4] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Interactive object counting," in *European Conference on Computer Vision*, 2014.

[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*, 2013.

[6] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, "Deep People Counting in Extremely Dense Crowds," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015.

[7] L. Fiaschi, R. Nair, U. Koethe, and F. A. Hamprecht, "Learning to count with regression forest and structured labels," in *International Conference on Pattern Recognition*, 2012.

[8] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[9] V.-Q. Pham, T. Kozakaya, O. Yamaguchi, and R. Okada, "COUNT Forest: CO-voting Uncertain Number of Targets using Random Forest for Crowd Density Estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[10] 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*, 2016.

[11] Z. Ma, L. Yu, and A. B. Chan, "Small instance detection by integer programming on object density maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[12] M. Rodriguez, I. Laptev, J. Sivic, and J.-Y. Audibert, "Density-aware person detection and tracking in crowds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[13] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting with fully convolutional regression networks," in *MICCAI 1st Workshop on Deep Learning in Medical Image Analysis*, 2015.

[14] D. Onoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *European Conference on Computer Vision*, 2016.

[15] C. Arteta, V. Lempitsky, and A. Zisserman, "Counting in The Wild," in *European Conference on Computer Vision*, 2016.

[16] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[17] G. J. Brostow and R. Cipolla, "Unsupervised bayesian detection of independent motion in crowds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[18] V. Rabaud and S. Belongie, "Counting crowded moving objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[19] D. Kong, D. Gray, and H. Tao, "A viewpoint invariant approach for crowd counting," in *International Conference on Pattern Recognition*, 2006.

[20] S.-Y. Cho, T. W. Chow, and C.-T. Leung, "A neural-based crowd estimation by hybrid global learning algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 4, 1999.

[21] K. Chen, S. Gong, T. Xiang, and C. Loy, "Cumulative attribute space for age and crowd density estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[22] K. Chen, C. C. Loy, S. Gong, and T. Xiang, "Feature mining for localised crowd counting," in *Proceedings of BMVC*, 2012.

[23] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and bayesian regression," *IEEE Transactions on Image Processing*, vol. 21, no. 4, 2012.

[24] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *European Conference on Computer Vision*, 2016.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[30] S. Li, Z.-Q. Liu, and A. B. Chan, "Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network," *International Journal of Computer Vision*, 2015.

[31] S. Li and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network," in *Asian Conference on Computer Vision*, 2014.

[32] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua, "Ordinal regression with multiple output cnn for age estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4920–4928.

[33] R. Rothe, R. Timofte, and L. Van Gool, "Dex: Deep expectation of apparent age from a single image," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 10–15.

[34] "Cs231n: Convolutional neural networks for visual recognition," http://cs231n.github.io/neural-networks-2/, 2018.

[35] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," in *International Conference on Learning Representations*, 2015.

[36] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[37] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, 2015.

[38] ——, "Exploiting the circulant structure of tracking-by-detection with kernels," in *European Conference on Computer Vision*, 2012.

[39] R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. Oñoro-Rubio, "Extremely overlapping vehicle counting," in *Pattern Recognition and Image Analysis*, 2015.

[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[41] S. An, W. Liu, and S. Venkatesh, "Face recognition using kernel ridge regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–7.

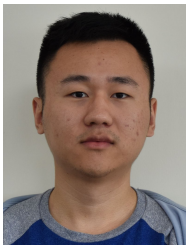
[42] P. Sudowe and B. Leibe, "Efficient use of geometric constraints for sliding-window object detection in video," in *Computer Vision Systems*, 2011.

[43] C. Arteta, V. Lempitsky, J. Noble, and A. Zisserman, "Learning to detect partially overlapping instances," in *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[44] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 2, 2017, p. 3.

[45] H. Li, R. Zhao, and X. Wang, "Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification," *arXiv preprint arXiv:1412.4526*, 2014.



Di Kang received the B.Eng. degree in Electronic Engineering and Information Science (EEIS) from the University of Science and Technology of China (USTC) in 2014. He is currently working towards the PhD degree in Computer Science at the City University of Hong Kong. His research interests include computer vision, crowd counting and deep learning.

Zheng Ma received the B.Eng. and M.Sc. from Xi'an Jiaotong University in 2007 and 2011, respectively, and the Ph.D. degree in computer science from City University of Hong Kong. Currently, he is a SenseTime research scientist. His research interests include computer vision, crowd counting, and object detection.

Antoni B. Chan received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, in 2000 and 2001, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, in 2008. In 2009, he was a Postdoctoral Researcher with the Statistical Visual Computing Laboratory, UCSD. In 2009, he joined the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, and is now serving as an Associate Professor. His research interests include computer vision, machine learning, pattern recognition, and music analysis. Dr. Chan was the recipient of an NSF IGERT Fellowship from 2006 to 2008, and an Early Career Award in 2012 from the Research Grants Council of the Hong Kong SAR, China.