Improving Dense Crowd Counting Convolutional Neural Networks using Inverse k-Nearest Neighbor Maps and Multiscale Upsampling

Greg Olmschenk[1], Hao Tang[2] and Zhigang Zhu[1,3]
[1]*The Graduate Center of the City University of New York, New York, USA*
[2]*Borough of Manhattan Community College - CUNY, New York, USA*
[3]*The City College of New York - CUNY, New York, USA*
golmschenk@gradcenter.cuny.edu, htang@bmcc.cuny.edu, zzhu@ccny.cuny.edu

Keywords: crowd counting, convolutional neural network, k-nearest neighbor, upsampling

Abstract: Gatherings of thousands to millions of people frequently occur for an enormous variety of events, and automated counting of these high-density crowds is useful for safety, management, and measuring significance of an event. In this work, we show that the regularly accepted labeling scheme of crowd density maps for training deep neural networks is less effective than our alternative inverse k-nearest neighbor (ikNN) maps, even when used directly in existing state-of-the-art network structures. We also provide a new network architecture MUD-ikNN, which uses multi-scale drop-in replacement upsampling via transposed convolutions to take full advantage of the provided ikNN labeling. This upsampling combined with the ikNN maps further improves crowd counting accuracy. Our new network architecture performs favorably in comparison with the state-of-the-art. However, our labeling and upsampling techniques are generally applicable to existing crowd counting architectures.

1 Introduction

Every year, gatherings of thousands to millions occur for protests, festivals, pilgrimages, marathons, concerts, and sports events. For any of these events, there are countless reasons to desire to know how many people are present. For those hosting the event, both real-time management and future event planning is dependent on how many people are present, where they are located, and when they are present. For security purposes, knowing how quickly evacuations can be executed and where crowding might pose a threat to individuals is dependent on the size of the crowds. In journalism, crowd sizes are frequently used to measure the significance of an event, and systems which can accurately report on the event size are important for a rigorous evaluation.

Many systems have been proposed for crowd counting purposes, with most recent state-of-the-art methods being based on convolutional neural networks (CNNs). To the best of our knowledge, every CNN-based dense crowd counting approach in recent years relies on using a density map of individuals, primarily with a Gaussian-based distribution of density values centered on individuals labeled in the ground truth images. Often, these density maps are generated with the Gaussian distribution kernel sizes being dependent on a k-Nearest Neighbor (kNN) distance to other individuals [Zhang et al., 2016]. In this work, we explain how this generally accepted density map labeling is lacking and how an alternative inverse kNN (ikNN) labeling scheme, which does not explicitly represent crowd density, provides improved counting accuracy. We will show how a single ikNN map provides information similar to the accumulation of many density maps with different Gaussian spreads, in a form which is better suited for neural network training. This labeling provides a significant gradient spatially across the entire label while still providing precise location information of individual pedestrians (with the only exception being exactly overlapping head labelings). We show that by simply replacing density map training in an existing state-of-the-art network with our ikNN map training, the testing accuracy of the network improves. This is the first major contribution of the paper.

Additionally, coupling multi-scale drop-in replacement upsampling with densely connected convolutional networks [Huang et al., 2017] and our proposed ikNN mapping, we provide a new network structure, MUD-ikNN, which performs favorably compared to existing state-of-the-art methods. Our network integrates multi-scale upsampling with transposed convolutions [Zeiler et al., 2010] to make effective use of the full ground truth label, particularly with respect to our

ikNN labeling scheme. The transposed convolutions are used to spatially upsample intermediate feature maps to the ground truth label map size for comparison. This approach provides several benefits. First, it allows the features of any layer to be used in the full map comparison, where many existing methods require a special network branch for this comparison. Notably, this upsampling, comparison, and following regression module can be used at any point in any CNN, with the only change being the parameters of the transposed convolution. This makes the module useful not only in our specific network structure, but also applicable in future state-of-the-art, general-purpose CNNs. Second, as this allows features which have passed through different levels of convolutions to be compared to the ground truth label map, this intrinsically provides a multi-scale comparison without any dedicated additional network branches, thus preventing redundant parameters which occur in separate branches. Third, because the transposed convolution can provide any amount of upsampling (with the features being used to specify the upsampling transformation), the upsampled size can be the full ground truth label size. In contrast, most existing works used a severely reduced size label map for comparison. These reduced sizes remove potentially useful training information. Although some recent works use full-size labels, they require specially crafted network architectures to accomplish this comparison. Our proposed upsampling structure can easily be added to most networks, including widely used general-purpose networks, such as DenseNet. This proposed network structure is the second major contribution of the paper.

Importantly, these contributions are largely complementary to, rather than alternatives to, existing approaches. Most approaches can easily replace their density label comparison with our proposed ikNN map comparison and upsampling map module, with little to no modification of the rest of their method or network architecture. As the ikNN label does not sum to the count, the ikNN label and map module should go hand-in-hand.

The paper is organized as follows. Section 2 discusses related work. Section 3 proposes our new network architecture for crowd counting, MUD-ikNN. Section 4 details the proposed k-nearest neighbor map labeling method and its justification. Section 5 presents experimental results on several crowd datasets and analyzes the findings. Section 6 provides a few concluding remarks.

2 Related Work

Many works use explicit detection of individuals to count pedestrians [Wu and Nevatia, 2005, Lin and Davis, 2010, Wang and Wang, 2011]. However, as the number of people in a single image increase and a scene becomes crowded, these explicit detection methods become limited by occlusion effects. Early works to solve this problem relied on global regression of the crowd count using low-level features [Chan et al., 2008, Chen et al., 2012, Chen et al., 2013]. While many of these methods split the image into a grid to perform a global regression on each cell, they still largely ignored detailed spatial information of pedestrian locations. [Lempitsky and Zisserman, 2010] introduced a method of counting objects using density map regression, and this technique was shown to be particularly effective for crowd counting by [Zhang et al., 2015]. Since then, to the best of our knowledge, every CNN-based crowd counting method in recent years has used density maps as a primary part of their cost function [Idrees et al., 2018, Sam et al., 2017, Sindagi and Patel, 2017, Zhang et al., 2015, Zhang et al., 2016, Shen et al., 2018, Li et al., 2018, Ranjan et al., 2018, Shi et al., 2018].

A primary advantage of the density maps is the ability to provide a useful gradient for network training over large portions of the image spatially, which helps the network identify which portion of the image contains information signifying an increase in the count. These density maps are usually modeled by representing each labeled head position with a Dirac delta function, and convolving this function with a 2D Gaussian kernel [Lempitsky and Zisserman, 2010]. This forms a density map where the sum of the total map is equal to the total count of individuals, while the density of a single individual is spread out over several pixels of the map. The Gaussian convolution allows a smoother gradient for the loss function of the CNN to operate over, thereby allowing slightly misplaced densities to result in a lower loss than significantly misplaced densities.

In some works, the spread parameter of the Gaussian kernel is often determined using a k-nearest neighbor (kNN) distance to other head positions [Zhang et al., 2016]. This provides a form of pseudo-perspective which results in pedestrians which are more distant from the camera (and therefore smaller in the image) having their density spread over a smaller number of density map pixels. While this mapping will often imperfectly map perspective (especially in sparsely crowded images), it works well in practice. Whether adaptively chosen or fixed, the

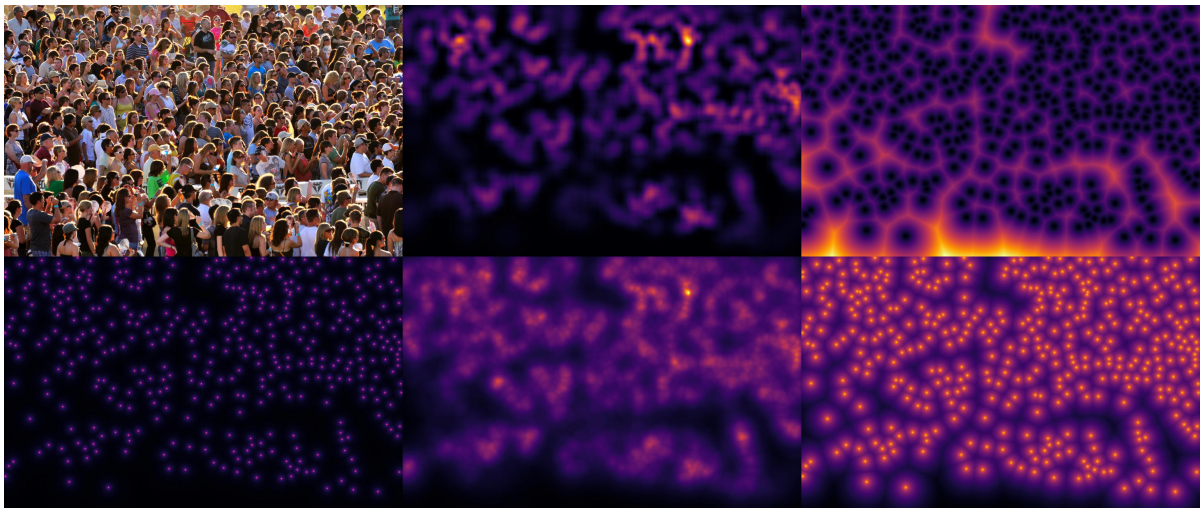


Figure 1: An example of a crowd image and various kinds of labelings. From left to right, on top: the original image, the density map, the kNN map with $k = 1$. On bottom: the inverse kNN map with $k = 1$, $k = 3$, and $k = 1$ shown with a log scaling (for reader insight only). Note, in the case of the density map, any values a significant distance from a head labeling are very small. In contrast, the inverse kNN map has a significant gradient even a significant distance from a head position.

Gaussian kernel size is dependent on arbitrarily chosen parameters, usually fine-tuned for a specific dataset. We compare with adaptive version in this work, due to its success and being more closely related to our method.

In a recent work [Idrees et al., 2018], the authors used multiple scales of these kNN-based, Gaussian convolved density maps to provide levels of spatial information, from large Gaussian kernels (allowing for a widespread training gradient) to small Gaussian kernels (allowing for precise localization of density). While this approach effectively integrates information from multiple Gaussian scales, thus providing both widespread and precise training information, the network is left with redundant structures and how the various scales are chosen is fairly ad hoc. Our alternative ikNN labeling method supersedes these multiple scale density maps by providing both a smooth training gradient and precise label locations (in the form of steep gradients) in a single label. Our new network structure utilizes a single branch CNN structure for multi-scale regression. Together with the ikNN labeling, it provides the benefits of numerous scales of these density maps.

Though most CNN-based approaches use a reduced label size, some recent works [Shen et al., 2018, Li et al., 2018, Cao et al., 2018, Laradji et al., 2018] have begun using full resolution labels. In contrast even to these works, we provide a generalized map module which can be added to existing network structures. Specifically, the map module can be used as a drop-in replacement for the density map comparisons. This map module can be added to most dense crowd counting architectures with little or no modification to the original architecture. In this paper, our proposed network is based off the DenseNet201 [Huang et al., 2017], with our map module added to the end of each DenseBlock.

Our ikNN mapping is obliquely related to a distance transform, which has been used for counting in other applications [Arteta et al., 2016]. However, the distance transform is analogous to a kNN map, rather than our ikNN. Notably, the ikNN crowd labeling presents the network with a variable training gradient to the network, with low values far from head labelings and cusps at a head labeling. In contrast, a kNN or distance transform provides constant training gradients everywhere. To our knowledge, neither the distance transform nor a method analogous to our ikNN labeling has been used for dense crowd counting.

3 MUD-ikNN: A New Network Architecture

We propose a new network structure, MUD-ikNN, with both multi-scale upsampling using DenseBlocks [Huang et al., 2017] and our ikNN mapping scheme. For providing a context of our proposed ikNN mapping scheme, we will describe the network structure first, before the detailed description of the ikNN mapping in Section 4). We show that the new MUD-ikNN structure performs favorably compared with existing state-of-the-art networks. In addition to the use

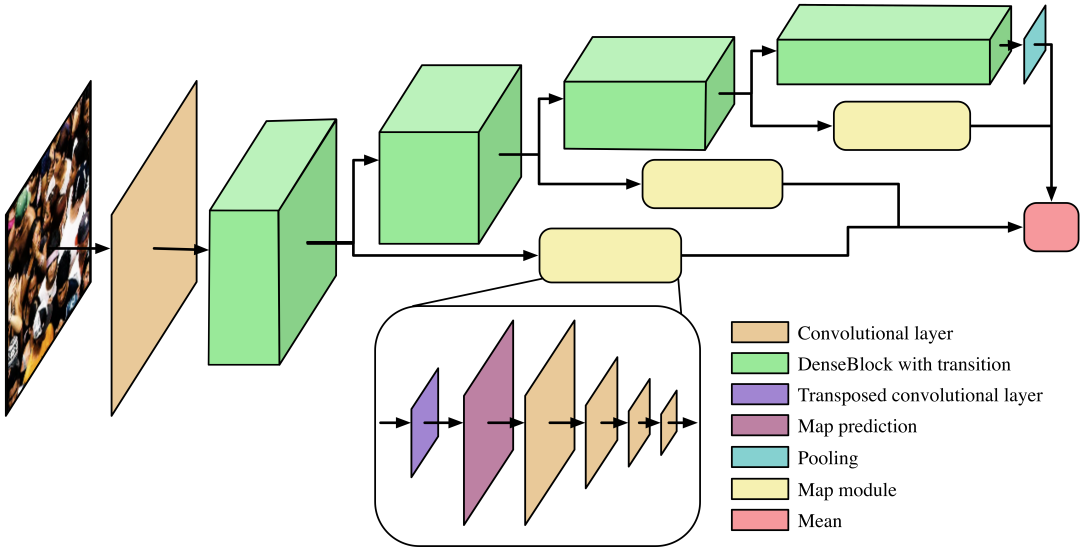


Figure 2: A diagram of the proposed network architecture MUD-ikNN: multiscale regression with DenseBlocks and ikNN mapping. Best viewed in color.

of ikNN maps playing a central role, we also demonstrate how features with any spatial size can contribute in the prediction of ikNN maps and counts through the use of transposed convolutions. This allows features of various scales from throughout the network to be used for the prediction of the crowd. Throughout this section, a "label map" may refer to either our ikNN map or a standard density map, as either can be used with our network.

The proposed MUD-ikNN network structure is shown in Figure 2. Our network uses the DenseBlock structures from DenseNet201 [Huang et al., 2017] in their entirety. DenseNet has been shown to be widely applicable to various problems. The output of each DenseBlock (plus transition layer) is used as the input to the following DenseBlock, just as it is in DenseNet201. However, each of these outputs is also passed to a map module (excluding the final DenseBlock output), which includes a transposed convolutional layer, a map prediction layer, and a small count regression module with four convolution layers. For each transposed convolution, the kernel size and stride are the same value, resulting in each spatial input element being transformed to multiple spatial output elements. The kernel size/stride value is chosen for each DenseBlock such that resulting map prediction is the size of the ground truth label. This form of upsampling using transposed convolutions allows the feature depth dimensions to contribute to the gradients of the map values in the predicted label map. Both the stride and kernel size of the transposed convolutions of our network are 8, 16, and 32 for the first three Denseblocks, respectively.

The label map generated at after each DenseBlock is individually compared against the ground truth label map, each producing a loss which is then summed,

$$\mathcal{L}_m = \sum_j \text{MSE}(\hat{M}_j, M_j) \tag{1}$$

where j is the index of the DenseBlock that the output came from, M is the ground truth label map, and $\hat{M}$ is the predicted map labeling.

Each predicted label map is then also used as the input to a small count regression module. This module is a series of four convolutional layers, shown in the inset of Figure 2. The sizes of these layers are specified in Table 1. The regression module then has a singleton output, corresponding to the predicted crowd count.

The mean of all predicted crowd counts from the regression modules, three in Figure 2, and the output of the final DenseBlock is used as the final count prediction.

$$\mathcal{L}_c = \text{MSE}\left(\frac{\hat{C}_{end} + \sum_{j=1}^{m} \hat{C}_j}{m+1}, C\right) \tag{2}$$

with C being the ground truth count, $\hat{C}_{end}$ being the regression count output by the final DenseBlock, and $\hat{C}_j$ being the count from the jth map regression module ($j = 1, 2, ..., m; m = 3$ in Figure 2). This results in a total loss given by $\mathcal{L} = \mathcal{L}_m + \mathcal{L}_c$.

This approach has multiple benefits. First, if an appropriately sized stride and kernel size are specified, the transposed convolutional layer followed by label

Layer	Output size	Filter
Input from DenseBlock	128x28x28 256x14x14 896x7x7	
Transposed convolution	1x224x224 (map prediction)	(8,16,32)x(8,16,32) stride=(8,16,32)
Convolution	8x112x112	2x2 stride=2
Convolution	16x56x56	2x2 stride=2
Convolution	32x28x28	2x2 stride=2
Convolution	1x1x1	28x28

Table 1: A specification of the map module layers. This module is used at 3 points throughout our network as shown in Figure 2, so the initial input size varies. However, the transposed convolution always produces a predicted map label which is uniform size (1x224x224).

map prediction to regression module can accept any sized input. This means this module of the network is very generalizable and can be applied to any CNN structure at any point in the network. For example, an additional DenseBlock could be added to either end of the DenseNet, and another of these map modules could be attached. Second, each label map is individually trained to improve the prediction at that layer, which provides a form of intermediate supervision, easing the process of training earlier layers in the network. At the same time, the final count is based on the mean values of the regression modules. This means that if any individual regression module produces more accurate results, its results can individually be weighted as being more important to the final prediction.

We note that the multiple Gaussian approach by [Idrees et al., 2018] has some drawbacks. The spread of the Gaussians, as well as the number of different density maps, is arbitrarily chosen. Additionally, without upsampling, a separate network branch is required to maintain spatial resolution. This results in redundant network parameters and a final count predictor which is largely unconnected to the map prediction optimization goal. Our upsampling approach allows the main network to retain a single primary branch and connects all the optimization goals tightly to this branch.

The input to the network is 224×224 image patches. At evaluation time, a 224×224 sliding window with a step size of 128 was used for each part of the test images, with overlapping predictions averaged.The label maps use the same size patches, and predictions from the network are of the same resolution. Each count regression module contains the same four layers, as specified in Table 1.

For each experiment, the network was trained for 10^5 training steps. The network was designed and training process carried out using PyTorch (v0.4.0). The network was trained on a Nvidia GTX 1080 Ti. Complete details of the network code and hyperparameters can be found at https://github.com/golmschenk/sr-gan.

4 Inverse k-Nearest Neighbor Map Labeling

We propose using full image size ikNN maps as an alternative labeling scheme from the commonly used density map explained in the Related Work in (Section 2). Formally, the commonly used density map [Idrees et al., 2018, Sam et al., 2017, Sindagi and Patel, 2017, Zhang et al., 2015, Zhang et al., 2016] is provided by,

$$D(\boldsymbol{x}, f(\cdot)) = \sum_{h=1}^{H} \frac{1}{\sqrt{2\pi} f(\sigma_h)} \exp\left(-\frac{(x - x_h)^2 + (y - y_h)^2}{2 f(\sigma_h)^2}\right), \tag{3}$$

where H is the total number of head positions for the example image, σ_h is a size determined for each head position (x_h, y_h) using the kNN distance to other heads positions (a fixed size is also often used), and f is a manually determined function for scaling σ_h to provide a Gaussian kernel size. We use this adaptive Gaussian label as the baseline in our experiments. For simplicity, in our work we define f as a simple scalar function given by $f(\sigma_h) = \beta\sigma_h$, with β being a hand-picked scalar. Though they both apply to head positions, the use of kNN for σ_h in the density map is not to be confused with the full kNN map used in our method, which is defined by,

$$K(\boldsymbol{x}, k) = \frac{1}{k} \sum \min_{k} \left(\sqrt{(x - x_h)^2 + (y - y_h)^2}, \forall \boldsymbol{h} \in \mathcal{H}\right), \tag{4}$$

where $\mathcal{H}$ is the list of all head positions. In other words, the kNN distance from each pixel, (x, y), to each head position, (x_h, y_h), is calculated.

To produce the inverse kNN (ikNN) map, we use,

$$M = \frac{1}{K(\boldsymbol{x}, k) + 1}, \tag{5}$$

where M is the resulting ikNN map, with the addition and inverse being applied element-wise.

To understand the advantage of an ikNN map over a density map, we can consider taking the generation

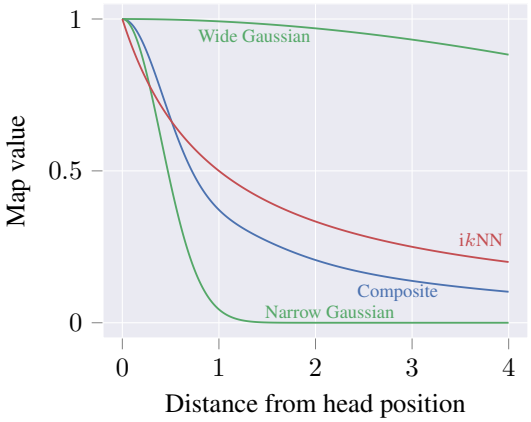


Figure 3: A comparison of the values of map labeling schemes with respect to the distance from an individual head position (normalized for comparison). Two Gaussians are shown in green. The narrow Gaussian provides a precise location of the head labeling. However, it provides little training information as the distance from the head increases. The wide Gaussian provides training information at a distance, but gives an imprecise location of the head position, resulting in low training information near the correct answer. The blue line shows a composite of several Gaussians with spread parameters between those of the two extremes ([Idrees et al., 2018] uses 3 Gaussian spreads in their work). This provides both precise and distant training losses. Our approach of the ikNN map shown in red (with $k = 1$) approaches a map function with a shape similar to the integral on the spread parameter of all Gaussians for a spread parameter range from 0 to some constant. Additionally, our method provides both the precise and distant gradient training information in a single map label. Also notable, is that even the large Gaussian shown here approaches near zero much sooner than the ikNN map value.

of density maps to extremes with regard to the spread parameter of the Gaussian kernel provided by f. A similar explanation is illustrated in Figure 3. At one extreme, is a Gaussian kernel with zero spread. Here the delta function remains unchanged, which in practical terms translates to a density map where the density for each pedestrian is fully residing on a single pixel. When the difference between the true and predicted density maps is used to calculate a training loss, the network predicting density 1 pixel away from the correct labeling is considered just as incorrect as 10 pixels away from the correct labeling. This is not desired, as it both creates a discontinuous training gradient, and the training process is intolerant to minor spatial labeling deviations. The other extreme is a very large Gaussian spread. This results in inexact spatial information of the location of the density. At the extreme, this provides no benefit over a global regression, which is the primary purpose for using a density map in the first place. The extreme cases are shown for explanatory purposes, yet any intermediate Gaussian spread has some degree of both these issue. Using multiple scales of Gaussian spread, [Idrees et al., 2018] tries to obtain the advantage of both sides. However, the size of the scales and the number of scales are then arbitrary and hard to determine.

In contrast, a single ikNN map provides a substantial gradient everywhere while still providing steep gradients in the exact locations of individual pedestrians. Notably, near zero distance, the ikNN mapping clearly has a greater slope, and in comparison, for any Gaussian there exists a distance at which all greater distances have a smaller slope than the equivalent position on the ikNN mapping. This means, the slope of the Gaussian is only greater than the slope of the ikNN mapping for an middle range arbitrarily determined by the Gaussian spread. The ikNN curve and its derivative's magnitude (the inverse distance squared) monotonically increase toward zero. We want to note here that directly using a kNN map doesn't have the advantage of using an inverse kNN map, since a kNN or distance transform provides constant training gradients everywhere. This was further verified in our preliminary experiments. An example of our ikNN map compared with a corresponding density map labeling can be seen in Figure 1. [Idrees et al., 2018] uses 3 density maps with different Gaussian spread parameters, with the Gaussian spread being determined by the kNN distance to other head positions multiplied by one of the 3 spread parameters. For a single head position, all Gaussian distributions integrated on β from 0 to an arbitrary constant results in a form of the incomplete gamma function. This function has a cusp around the center of the Gaussians. Similarly, the inverse of the kNN map also forms a cusp at the head position and results in similar gradients at corresponding distances as the integrated Gaussian function. In our experiments, we found that an inverse kNN map outperformed density maps with ideally selected spread parameters.

In one experiment, we use [Idrees et al., 2018]'s network architecture, which utilized DenseBlocks [Huang et al., 2017] as the basis, but we replace the density maps with ikNN maps and show there is an improvement in the prediction's mean absolute error. This demonstrates the direct improvement of our ikNN method on an existing state-of-the-art network. Note, the regression module from ikNN map to count is then also required to convert from the ikNN map to a count. The difference in error between the original approach in [Idrees et al., 2018] and the network in [Idrees et al., 2018] with our ikNN maps, though improved, is relatively small. We suspect this is because the density maps (or ikNN maps) used during training are downsampled to a size of 28x28

(where the original images and corresponding labels are 224x224). This severe downsampling results in more binning of pixel information, and this seems to reduce the importance of which system is used to generate that label. At the extreme case, when downsampled to a single value, both approaches would only give the global count in the patch (where the ikNN map gives the inverse of the average distance from a pixel to a head labeling which can be translated to an approximate count). This downsampling is a consequence of the network structure only permitting labels of the same spatial size as the output of the DenseBlocks. Our network (which will be described below) remedies this through transposed convolutions, allowing for the use of the full-size labels.

The generation of the ikNN labels occurs as one-time data preprocessing step before the training process, and thus the label generation method does not have an impact on the speed of training steps.

5 Experimental Results

5.1 Evaluation metrics

For each dataset that we evaluated our method on, we provide the mean absolute error (MAE), normalized absolute error (NAE), and root mean squared error (RMSE). These are given by the following equations:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{C}_i - C_i\right| \tag{6}$$

$$\text{NAE} = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\hat{C}_i - C_i\right|}{C_i} \tag{7}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{C}_i - C_i)^2} \tag{8}$$

In the first set of experiments, we demonstrate the improvement of the ikNN labeling scheme compared to the density labeling scheme. We trained our network using various density maps produced with different Gaussian spread parameters, β (as described in Section 4), and compared these results to the network using ikNN maps with varying k. We also analyze the advantage of upsampling the label for both density and ikNN maps. In the second set of experiments, we provide comparisons to the state-of-the-art on standard crowd counting datasets. In these comparisons, the best ikNN map and density map from the first set of experiments is used. Most works provide their MAE and RMSE results. [Idrees et al., 2018] provided the additional metric of NAE. Though this result is not available for many of the datasets, we provide our own NAE on these datasets for future works to refer to. The most directly relevant work, [Idrees et al., 2018], has only provided their results for their latest dataset, UCF-QNRF. As such, their results only appear in regard to that dataset. Finally, we offer a general analysis of the results using our ikNN maps and upsampling approaches. General statistics about the datasets used in our experiments is shown in Table 2.

5.2 Impact of labeling approach and upsampling

5.2.1 Density maps vs ikNN maps

We used the ShanghaiTech dataset [Zhang et al., 2016] part A for this analysis. The results of these tests are shown in Table 3. The density maps provide a curve, where too large and too small of spreads perform worse than an intermediate value. Even when choosing the best value (where $\beta = 0.3$), which needs to manually determined, the i1NN label significantly outperforms the density label.

Included in the table are experiments, in the fashion of [Idrees et al., 2018], with density maps using 3 different β values. Here β_1 denotes the spread parameter used as the label map for the first map module, while β_2 and β_3 are for the second and third modules. Contrary to [Idrees et al., 2018]'s findings, we only gained a benefit from 3 density labels when the first output had the smallest spread parameter. Even then, the gain was minimal. Upon inspection of the weights produced by the network from the map to the count prediction, the network reduces the predictions from the non-optimal β maps to near zero and relies solely on the optimal map (resulting in a reduced accuracy compared to using the optimal map for each map module).

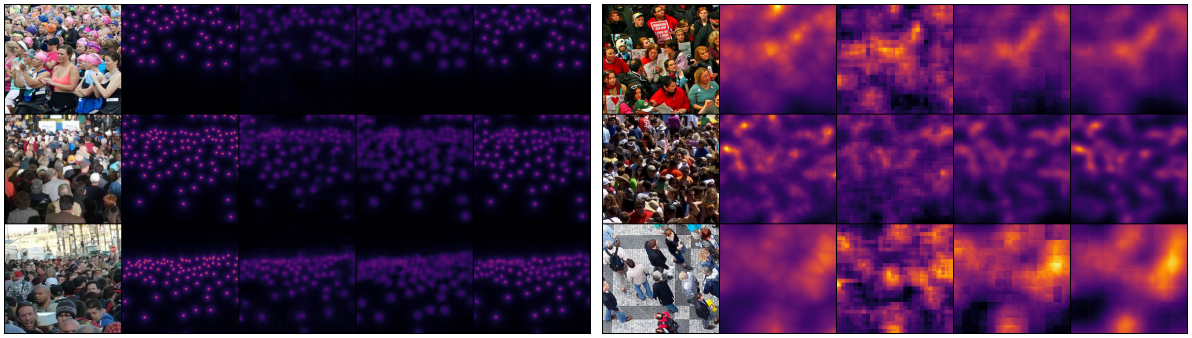
With varying k, we find that an increased k results in lower accuracy. This is likely due to the loss of precision in the location of an individual. The most direct explanation for this can be seen in the case of $k = 2$. Every pixel on the line between two nearest head positions will have the same map value, thus losing the precision of an individual location.

5.2.2 Upsampling analysis

Most existing works use a density map with a reduced size label for testing and training. Those that use the full label resolution design specific network architectures for the high-resolution labels. Our map module avoids this constraint by upsampling the label using

Dataset	Images	Total count	Mean count	Max count	Average resolution
UCF-QNRF	1535	1,251,642	815	12,865	2013×2902
ShanghaiTech Part A	482	241,677	501	3139	589×868
ShanghaiTech Part B	716	88,488	123.6	578	768×1024
UCF-CC-50	50	63,974	1279	4633	2101×2888

Table 2: General statistics for the tested datasets.



(a) i1NN predictions. (b) i3NN predictions.

Figure 4: A small sample of patch predictions for map labels. In each subfigure, from left to right is the original image patch, the ground truth label, and the patches from the three map modules in order through the network.

a trained transposed convolution, which can be integrated into most existing architectures. Using the ShanghaiTech part A dataset, we tested our network using various label resolutions to determine the impact on the predictive abilities of the network. These results can be seen in Table 3. Experiments without no label resolution given are 224×224. From these results, it is clear that the higher resolution leads to higher accuracy. Note, this results in a minor change to the map module structure, as the final convolution kernel needs to match the remaining spatial dimension. A set of predicted ikNN map labels can be seen in Figure 4, where a grid pattern due to the upsampling can be identified in some cases.

5.3 Comparisons on standard datasets

The following demonstrates our network's predictive capabilities on various datasets, compared to various state-of-the-art methods. Again, we note that our improvements are expected to complementary to the existing approaches, rather than alternatives.

For these experiments, we used the best k, 1, and best β, 0.3, from the first set of experiments.

The first dataset we evaluated our approach on is the UCF-QNRF dataset [Idrees et al., 2018]. The results of our MUD-ikNN network compared with other state-of-the-art networks are shown in Table 4. Our network significantly outperforms the existing methods. Along with a comparison of our complete method compared with the state-of-the-art, we compare with [Idrees et al., 2018]'s network, but replace their density map predictions and summing to count with our ikNN map prediction and regression to count. Using the ikNN maps, we see that their model sees improvement in MAE with ikNN maps, showing the effect of the ikNN mapping.

The second dataset we evaluated our approach on is the ShanghaiTech dataset [Zhang et al., 2016]. The dataset is split into two parts, Part A and Part B. For both parts, we used the training and testing images as prescribed by the dataset provider. The results of our evaluation on part A are shown in Table 5. Our MUD-ikNN network slightly outperforms the state-of-the-art approaches on this part. The results of our evaluation on part B are shown in Table 6. Here our network performs on par or slightly worse than the best-performing methods. Notably, our method appears perform better on denser crowd images, and ShanghaiTech Part B is by far the least dense dataset we tested.

The third dataset we evaluated our approach on is the UCF-CC-50 dataset [Idrees et al., 2013]. We followed the standard evaluation metric for this dataset of a five-fold cross-evaluation. The results of our evaluation on this dataset can be seen in Table 7.

Overall, our network performed favorably compared with existing approaches. An advantage to our approach is that the our modifications can be applied to the architectures we're comparing against. The most relevant comparison is between the ikNN version of the MUD network, and the density map version. Here, the ikNN approach always outperformed the density

Method	MAE	NAE	RMSE
MUD-densityβ0.3 28x28	79.0	0.209	120.5
MUD-densityβ0.3 56x56	74.8	0.181	121.0
MUD-densityβ0.3 112x112	73.3	0.176	119.1
MUD-i1NN 28x28	75.8	0.180	120.3
MUD-i1NN 56x56	72.7	0.181	117.4
MUD-i1NN 112x112	70.8	0.166	117.0
MUD-densityβ0.05	84.5	0.233	139.9
MUD-densityβ0.1	76.8	0.189	120.3
MUD-densityβ0.2	75.3	0.175	124.2
MUD-densityβ0.3	72.7	0.174	120.4
MUD-densityβ0.4	75.7	0.176	130.5
MUD-densityβ0.5	76.3	0.182	130.0
MUD-density $\beta_1$0.5,$\beta_2$0.3,$\beta_3$0	78.5	0.205	124.2
MUD-density $\beta_1$0.5,$\beta_2$0.3,$\beta_3$0.05	77.8	0.207	124.9
MUD-density $\beta_1$0.4,$\beta_2$0.2,$\beta_3$0.1	76.7	0.202	122.7
MUD-density $\beta_1$0.1,$\beta_2$0.2,$\beta_3$0.4	75.1	0.191	119.0
MUD-density $\beta_1$0.2,$\beta_2$0.3,$\beta_3$0.4	76.0	0.196	122.1
MUD-i1NN	68.0	0.162	117.7
MUD-i2NN	68.8	0.168	109.0
MUD-i3NN	69.8	0.169	110.7
MUD-i4NN	72.2	0.173	116.0
MUD-i5NN	74.0	0.182	119.1
MUD-i6NN	76.2	0.188	120.9

Table 3: Results using density maps vs ikNN maps with varying k and β, as well as the various upsampling resolutions on the ShanghaiTech Part A dataset. If a resolution is not shown, it is the default 224×224. Multiple β correspond to a different Gaussian density map for each of the 3 map module comparisons.

version. We speculate that the state-of-the-art methods we have compared with, along with other general-purpose CNNs, could be improved through the use of ikNN labels and upsampling map modules.

6 Conclusions

We have presented a new form of labeling for crowd counting data, the ikNN map. We have compared this labeling scheme to commonly accepted labeling approach for crowd counting, the density map. We show that using the ikNN map with an existing state-of-the-art network improves the accuracy of the network compared to density map labelings. We have demonstrated the improvements gained by using increased label resolutions, and provide an upsampling map module which can be generally used by other crowd counting architectures. These approaches can be used a drop-in replacement in other crowd counting architectures, as we have done for DenseNet, which resulted in a network which performs favorably compared with the state-of-the-art.

7 Acknowledgments

The research is supported by National Science Foundation through Awards PFI #1827505 and SCC-Planning #1737533, and Bentley Systems, Incorporated, through a CUNY-Bentley Collaborative Research Agreement (CRA). Additional support is provided by the Intelligence Community Center of Academic Excellence (IC CAE) at Rutgers University.

REFERENCES

[Arteta et al., 2016] Arteta, C., Lempitsky, V., and Zisserman, A. (2016). Counting in the wild. In *European conference on computer vision*, pages 483–498. Springer. 3

[Badrinarayanan et al., 2017] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495. 10

[Cao et al., 2018] Cao, X., Wang, Z., Zhao, Y., and Su, F. (2018). Scale aggregation network for accurate and efficient crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750. 3

[Chan et al., 2008] Chan, A. B., Liang, Z.-S. J., and Vasconcelos, N. (2008). Privacy preserving crowd monitoring: Counting people without people models or tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE. 2

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

[Chen et al., 2012] Chen, K., Loy, C. C., Gong, S., and Xiang, T. (2012). Feature mining for localised crowd counting. In *BMVC*, volume 1, page 3. 2

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 10

Method	MAE	NAE	RMSE
Idrees *et al.*(2013) [Idrees et al., 2013]	315	0.63	508
MCNN [Zhang et al., 2016]	277	0.55	426
Encoder-Decoder [Badrinarayanan et al., 2017]	270	0.56	478
CMTL [Sindagi and Patel, 2017]	252	0.54	514
SwitchCNN [Sam et al., 2017]	228	0.44	445
Resnet101 [He et al., 2016]	190	0.50	227
DenseNet201 [Huang et al., 2017]	163	0.40	226
Idrees *et al.*(2018) [Idrees et al., 2018]	132	0.26	191
[Idrees et al., 2018] with i1NN maps	122	0.252	195
MUD-i1NN	104	0.209	172

Table 4: Results on the UCF-QNRF dataset.

Method	MAE	NAE	RMSE
ACSCP [Shen et al., 2018]	75.7	-	102.7
D-ConvNet-v1 [Shi et al., 2018]	73.5	-	112.3
ic-CNN [Ranjan et al., 2018]	68.5	-	116.2
CSRNet [Li et al., 2018]	68.2	-	115.0
MUD-densityβ0.3	72.7	0.174	120.4
MUD-i1NN	68.0	0.162	117.7

Table 5: Results on the ShanghaiTech Part A dataset.

Method	MAE	NAE	RMSE
D-ConvNet-v1 [Shi et al., 2018]	18.7	-	26.0
ACSCP [Shen et al., 2018]	17.2	-	27.4
ic-CNN [Ranjan et al., 2018]	10.7	-	16.0
CSRNet [Li et al., 2018]	10.6	-	16.0
MUD-densityβ0.3	16.6	0.130	26.9
MUD-i1NN	13.4	0.107	21.4

Table 6: Results on the ShanghaiTech Part B dataset.

Method	MAE	NAE	RMSE
ACSCP [Shen et al., 2018]	291.0	-	404.6
D-ConvNet-v1[Shi et al., 2018]	288.4	-	404.7
CSRNet [Li et al., 2018]	266.1	-	397.5
ic-CNN [Ranjan et al., 2018]	260.9	-	365.5
MUD-densityβ0.3	246.44	0.188	348.1
MUD-i1NN	237.76	0.191	305.7

Table 7: Results on the UCF-CC-50 dataset.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *CVPR*, volume 1, page 3. 1, 3, 4, 6, 10

[Idrees et al., 2013] Idrees, H., Saleemi, I., Seibert, C., and Shah, M. (2013). Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2547–2554. 8, 10

[Idrees et al., 2018] Idrees, H., Tayyab, M., Athrey, K., Zhang, D., Al-Maadeed, S., Rajpoot, N., and Shah, M. (2018). Composition loss for counting, density map estimation and localization in dense crowds. *arXiv preprint arXiv:1808.01050*. 2, 3, 5, 6, 7, 8, 10

[Laradji et al., 2018] Laradji, I. H., Rostamzadeh, N., Pinheiro, P. O., Vazquez, D., and Schmidt, M. (2018). Where are the blobs: Counting by localization with point supervision. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 547–562. 3

[Lempitsky and Zisserman, 2010] Lempitsky, V. and Zisserman, A. (2010). Learning to count objects in images. In *Advances in neural information processing systems*, pages 1324–1332. 2

[Li et al., 2018] Li, Y., Zhang, X., and Chen, D. (2018). Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1091–1100. 2, 3, 10

[Lin and Davis, 2010] Lin, Z. and Davis, L. S. (2010). Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):604–618. 2

[Ranjan et al., 2018] Ranjan, V., Le, H., and Hoai, M. (2018). Iterative crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–285. 2, 10

[Sam et al., 2017] Sam, D. B., Surya, S., and Babu, R. V. (2017). Switching convolutional neural network for crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 6. 2, 5, 10

[Shen et al., 2018] Shen, Z., Xu, Y., Ni, B., Wang, M., Hu, J., and Yang, X. (2018). Crowd counting via adversarial cross-scale consistency pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5245–5254. 2, 3, 10

[Shi et al., 2018] Shi, Z., Zhang, L., Liu, Y., Cao, X., Ye, Y., Cheng, M.-M., and Zheng, G. (2018). Crowd counting with deep negative correlation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5382–5390. 2, 10

[Sindagi and Patel, 2017] Sindagi, V. A. and Patel, V. M. (2017). Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, pages 1–6. IEEE. 2, 5, 10

[Wang and Wang, 2011] Wang, M. and Wang, X. (2011). Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3401–3408. IEEE. 2

[Wu and Nevatia, 2005] Wu, B. and Nevatia, R. (2005). Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *null*, pages 90–97. IEEE. 2

[Zeiler et al., 2010] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE. 1

[Zhang et al., 2015] Zhang, C., Li, H., Wang, X., and Yang, X. (2015). Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 833–841. 2, 5

[Zhang et al., 2016] Zhang, Y., Zhou, D., Chen, S., Gao, S., and Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597. 1, 2, 5, 7, 8, 10