Abstract

Automatic estimation of the number of people in unconstrained crowded scenes is a challenging task and one major difficulty stems from the huge scale variation of people. In this paper, we propose a novel Deep Structured Scale Integration Network (DSSINet) for crowd counting, which addresses the scale variation of people by using structured feature representation learning and hierarchically structured loss function optimization. Unlike conventional methods which directly fuse multiple features with weighted average or concatenation, we first introduce a Structured Feature Enhancement Module based on conditional random fields (CRFs) to refine multiscale features mutually with a message passing mechanism. In this module, each scale-specific feature is considered as a continuous random variable and passes complementary information to refine the features at other scales. Second, we utilize a Dilated Multiscale Structural Similarity loss to enforce our DSSINet to learn the local correlation of people’s scales within regions of various size, thus yielding high-quality density maps. Extensive experiments on four challenging benchmarks well demonstrate the effectiveness of our method. Specifically, our DSSINet achieves improvements of 9.5% error reduction on Shanghaitech dataset and 24.9% on UCF-QNRF dataset against the state-of-the-art methods.

1. Introduction

Crowd counting, which aims to automatically generate crowd density maps and estimate the number of people in unconstrained scenes, is a crucial research topic in computer vision. Recently, it has received increasing interests in both academic and industrial communities, due to its wide range of practical applications, such as video surveillance [40], traffic management [46] and traffic forecast [21].

Although numerous models [8, 11] have been proposed, crowd counting remains a very challenging problem and one major difficulty originates from the huge scale variation of people. As shown in Fig. 1, the scales of people vary greatly in different crowded scenes and capturing such huge scale variation is non-trivial. Recently, deep neural networks have been widely used in crowd counting and have made substantial progress [43, 25, 42, 24, 45, 22, 27]. To address scale variation, most previous works utilized mult-column CNN [47] or stacked multi-branch blocks [4] to extract multiple features with different receptive fields, and then fused them with weighted average or concatenation. However, the scale variation of people in diverse scenarios is still far from being fully solved.

In order to address scale variation, our motivations are two-fold. First, features of different scales contain different information and are highly complementary. For instance, the features at deeper layers encode high-level semantic information, while the features at shallower layers contain more low-level appearance details. Some related researches [41, 44] have shown that these complementary features can be mutually refined and thereby become more robust to scale variation. However, most existing methods use simple strategies (e.g., weighted average and concatenation) to fuse multiple features, and can not well capture the complementary information. Therefore, it is very necessary to propose
Second, crowd density maps contain rich information of the people’s scale, which can be captured by an effective loss function. As shown in Fig. 2, people in a local region usually have similar scales and the radiiuses of their heads are relatively uniform on density maps, which refers to the local correlation of people’s scales in our paper. Moreover, this pattern may vary in different locations. i) In the area near the camera, the radiiuses of people’s heads are large and their density values are consistently low, thus it’s better to capture the local correlation of this case in a large region (See the green box in Fig. 2). ii) In the place far away from the camera, the heads’ radiiuses are relatively small and the density map is sharp, thus we could capture the local correlation of this case in a small region (See the red box in Fig. 2). However, the commonly used pixel-wise Euclidean loss fails to adapt to these diverse patterns. Therefore, it is desirable to design a structured loss function to model the local correlation within regions of different sizes.

In this paper, we propose a novel Deep Structured Scale Integration Network (DSSINet) for high-quality crowd density maps generation, which addresses the scale variation of people from two aspects, including structured feature representation learning and hierarchically structured loss function optimization. First, our DSSINet consists of three parallel subnetworks with shared parameters, and each of them takes a different scaled version of the same image as input for feature extraction. Then, a unified Structured Feature Enhancement Module (SFEM) is proposed and integrated into our network for structured feature representation learning. Based on the conditional random fields (CRFs) [16], SFEM mutually refines the multiscale features from different subnetworks with a message passing mechanism [15]. Specifically, SFEM dynamically passes the complementary information from the features at other scales to enhance the scale-specific feature. Finally, we generate multiple side output density maps from the refined features and obtain a high-resolution density map in a top-down manner.

For the hierarchically structured loss optimization, we utilize a Dilated Multiscale Structural Similarity (DMS-SSIM) loss to enforce networks to learn the local correlation within regions of various sizes and produce locally consistent density maps. Specifically, our DMS-SSIM loss is designed for each pixel and it is computed by measuring the structural similarity between the multiscale regions centered at the given pixel on an estimated density map and the corresponding regions on ground-truth. Moreover, we implement the DMS-SSIM loss with a dilated convolutional neural network, in which the dilated operation enlarges the diversity of the scales of local regions and can further improve the performance.

In summary, the contributions of our work are three-fold:

- We propose a CRFs-based Structured Feature Enhancement Module, which refines multiscale features mutually and boosts their robustness against scale variation by fully exploiting their complementarity.
- We utilize a Dilated Multiscale Structural Similarity loss to learn the local correlation within regions of various sizes. To our best knowledge, we are the first to incorporate the MS-SSIM [39] based loss function for crowd counting and verify its effectiveness in this task.
- Extensive experiments conducted on four challenging benchmarks demonstrate that our method achieves state-of-the-art performance.

2. Related Work

Crowd Counting: Numerous deep learning based methods [36, 30, 40, 19, 20, 23, 7] have been proposed for crowd counting. These methods have various network structures and the mainstream is a multiscale architecture, which extracts multiple features from different columns/branches of networks to handle the scale variation of people. For instance, Boominathan et al. [3] combined a deep network and a shallow network to learn scale-robust features. Zhang et al. [47] developed a multi-column CNN to generate density maps. HydraCNN [25] fed a pyramid of image patches into networks to estimate the count. CP-CNN [35] proposed a Contextual Pyramid CNN to incorporate the global and local contextual information for crowd counting. Cao et al. [4] built an encoder-decoder network with multiple scale aggregation modules. However, the issue of the huge variation of people’s scales is still far from being fully solved. In this paper, we further strengthen the robustness of DSSINet against the scale variation of people from two aspects, including structured feature representation learning and hierarchically structured loss function optimization.

Conditional Random Fields: In the field of computer vision, CRFs have been exploited to refine the features and outputs of convolutional neural networks (CNN) with a message passing mechanism [15]. For instance, Zhang et

Figure 2. Illustration of the information of people’s scales on crowd density maps. The radiiuses of people’s heads are relatively uniform in a local region. Moreover, the local correlation of people’s scales may vary in different regions.
al. [49] used CRFs to refine the semantic segmentation maps of CNN by modeling the relationship among pixels. Xu et al. [41] fused multiple features with Attention-Gated CRFs to produce richer representations for contour prediction. Wang et al. [37] introduced an inter-view message passing module based on CRFs to enhance the view-specific features for action recognition. In crowd counting, we are the first to utilize CRFs to mutually refine multiple features at different scales and prove its effectiveness for this task.

Multiscale Structural Similarity: MS-SSIM [39] is a widely used metric for image quality assessment. Its formula is based on the luminance, contrast and structure comparisons between the multiscale regions of two images. In [48], MS-SSIM loss has been successfully applied in image restoration tasks (e.g., image denoising and super-resolution), but its effectiveness has not been verified in high-level tasks (e.g., crowd counting). Recently, Cao et al. [4] combined Euclidean loss and SSIM loss [38] to optimize their network for crowd counting, but they can only prove the scale variation of people with structured random fields (CRFs) is a widely used metric for image quality assessment. Its formula is based on the luminance, contrast and structure comparisons between the multiscale regions of two images. In [48], MS-SSIM loss has been successfully applied in image restoration tasks (e.g., image denoising and super-resolution), but its effectiveness has not been verified in high-level tasks (e.g., crowd counting). Recently, Cao et al. [4] combined Euclidean loss and SSIM loss [38] to optimize their network for crowd counting, but they can only capture the local correlation in regions with a fixed size. In this paper, to learn the local correlation within regions of various sizes, we modify MS-SSIM loss with a dilated operation and show its effectiveness in this high-level task.

3. Method

In this section, we propose a Deep Structured Scale Integration Network (DSSINet) for crowd counting. Specifically, it addresses the scale variation of people with structured feature representation learning and structured loss function optimization. For the former, a Structured Feature Enhancement Module based on conditional random fields (CRFs) is proposed to refine multiscale features mutually with a message passing mechanism. For the latter, a Dilated Multiscale Structural Similarity loss is utilized to enforce networks to learn the local correlation within regions of various sizes.

3.1. DSSINet Overview

In crowd counting, multiscale features are usually extracted to handle the scale variation of people. Inspired by [18, 43, 5], we build our DSSINet with three parallel subnetworks, which have the same architecture and share parameters. As shown in Fig. 3, these subnetworks are composed of the first ten convolutional layers (up to Conv4_3 of VGG16 [33] and each of them takes a different scaled version of the same image as input to extract features. Unlike previous works [47, 4] that simply fuse features by weighted averaging or concatenation, our DSSINet adequately refines the multiscale features from different subnetworks.

Given an image of size $H \times W$, we first build a three-levels image pyramid $\{I_1, I_2, I_3\}$, where $I_2 \in R^{H \times W}$ is the original image, and $I_1 \in R^{2H \times 2W}$ and $I_3 \in R^{H/4 \times W/4}$ are the scaled ones. Each of these images is fed into one of the subnetworks. The feature of image $I_k$ at the Conv_i,j layer of VGG16 is denoted as $f_{i,j}^k$. We then group the features with the same resolution from different subnetworks and form four sets of multiscale features $\{f_{1,2}^1, f_{1,2}^2, f_{1,2}^3\}$. $\{f_{3,3}, f_{2,2}, f_{1,1}\}$, $\{f_{1,3}, f_{3,3}, f_{2,2}\}$, $\{f_{1,2}, f_{3,3}\}$. In each set, different features complement each other, because they are inferred from different receptive fields and are derived from different convolutional layers of various image sizes. For instance, $f_{1,2}^2$ mainly contains the appearance details and $f_{1,2}^2$ encodes some high-level semantic information. To improve the robustness of scale variation, we refine the features in the aforementioned four sets with the Structured Feature Enhancement Module described in Section 3.2.

With richer information, the enhanced feature $f_{i,j}^k$ of $f_{i,j}$ becomes more robust to the scale variation. Then, $f_{i,j}^k$ is fed into the following layer of $L$-th subnetwork for deeper feature representation learning.

After structured feature learning, we generate a high-

Figure 3. The overall framework of the proposed Deep Structured Scale Integration Network (DSSINet). DSSINet consists of three parallel subnetworks with shared parameters. These subnetworks take different scaled versions of the same image as input for feature extraction. First, DSSINet integrates four CRFs-based Structured Feature Enhancement Modules (SFEM) to refine the multiscale features from different subnetworks. Then, we progressively fuse multiple side output density maps to obtain a high-resolution one in a top-down manner. Finally, a Dilated Multiscale Structural Similarity (DMS-SSIM) loss is utilized to optimize our DSSINet.
resolution density map in a top-down manner. First, we apply a \(1 \times 1\) convolutional layer on top of the last feature \(f^3\) for reducing its channel number to 128, and then feed the compressed feature into a \(3 \times 3\) convolutional layer to generate a density map \(M_i\). However, with a low resolution of \(\frac{H}{2^i} \times \frac{W}{2^i}\), \(M_i\) lacks spatial information of people. To address this issue, we generate other four side output density maps \(M_0, M_1, M_2, M_3\) at shallower layers, where \(M_i\) has a resolution of \(\frac{H}{2^i} \times \frac{W}{2^i}\). Specifically, \(M_3\) is computed by feeding the concatenation of \(f^2_1\) and \(f^3_3\) into two stacked convolutional layers. The first \(1 \times 1\) convolutional layer is also utilized to reduce the channel number of the concatenated feature to 128, while the second \(3 \times 3\) convolutional layer is used to regress \(M_3, M_2, M_1, M_0\) in the same manner. Finally, we progressively pass the density maps at deeper layers to refine the density maps at shallower layers and the whole process can be expressed as:

\[
M_i = w_i \ast M_i + w_{i+1} \ast U\hat{p}(M_{i+1}), \quad i = 3, 2, 1, 0 \tag{1}
\]

where \(w_i\) and \(w_{i+1}\) are the parameters of two \(3 \times 3\) convolutional layers and \(U\hat{p}(\cdot)\) denotes a bilinear interpolation operation with a upsampling rate of 2. \(M_i\) is the refined density map of \(M_i\). The final crowd density map \(M_0 \in \mathbb{R}^{H \times W}\) has fine details of the spatial distribution of people.

Finally, we train our DSSINet with the Dilated Multiscale Structural Similarity (DMS-SSIM) loss described in Section 3.3. We implement DMS-SSIM loss with a lightweight dilated convolutional neural network with fixed Gaussian-kernel and the gradient can be back-propagated to optimize our DSSINet.

### 3.2. Structured Feature Enhancement Module

In this subsection, we propose a unified Structured Feature Enhancement Module (SFEM) to improve the robustness of our feature for scale variation. Inspired by the dense prediction works [6, 41], our SFEM mutually refines the features at different scales by fully exploring their complementarity with a conditional random fields (CRFs) model. In this module, each scale-specific feature passes its own information to features at other scales. Meanwhile, each feature is refined by dynamically fusing the complementary information received from other features.

Let us denote multiple features extracted from different subnetworks as \(F = \{f_1, f_2, \ldots, f_n\}\). \(F\) can be any of the multiscale features sets defined in Section 3.1. Our objective is to estimate a group of refined features \(\hat{F} = \{\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n\}\), where \(\hat{f}_i\) is the corresponding refined feature of \(f_i\). We formulate this problem with a CRFs model. Specifically, the conditional distribution of the original feature \(F\) and the refined feature \(\hat{F}\) is defined as:

\[
P(F|\Theta) = \frac{1}{Z(F)} \exp\{E(\hat{F}, F, \Theta)\}, \tag{2}
\]

where \(Z(F) = \int_{\hat{F}} \exp\{E(\hat{F}, F, \Theta)\} d\hat{F}\) is the partition function for normalization and \(\Theta\) is the set of parameters. The energy function \(E(\hat{F}, F, \Theta)\) in CRFs is defined as:

\[
E(\hat{F}, F, \Theta) = \sum_i \phi(\hat{f}_i, f_i) + \sum_{i,j} \psi(\hat{f}_i, \hat{f}_j). \tag{3}
\]

In particular, the unary potential \(\phi(\hat{f}_i, f_i)\), indicating the similarity between the original feature and the refined feature, is defined as:

\[
\phi(\hat{f}_i, f_i) = -\frac{1}{2} ||\hat{f}_i - f_i||^2. \tag{4}
\]

We model the correlation between two refined features with a bilinear potential function, thus the pairwise potential is defined as:

\[
\psi(\hat{f}_i, \hat{f}_j) = (\hat{f}_i)^T w_j \hat{f}_j, \tag{5}
\]

where \(w_j\) is a learned parameter used to compute the relationship between \(\hat{f}_i\) and \(\hat{f}_j\).

This is a typical formulation of CRF and we solve it with mean-field inference [29]. The feature \(\hat{f}_i\) is computed by:

\[
\hat{f}_i = f_i + \sum_{j \neq i} w_j \hat{f}_j, \tag{6}
\]

where the unary term is feature \(f_i\) itself and the second term denotes the information received from other features at different scales. The parameter \(w_j\) determines the information content passed from \(f_j\) to \(f_i\). As \(\hat{f}_i\) and \(\hat{f}_j\) are inter-dependent in Eq. 5, we obtain each refined feature iteratively with the following formulation:

\[
h_i^0 = f_i, \quad h_i^t = f_i + \sum_{j \neq i} w_j^t h_j^{t-1}, t = 1 to n, \quad \hat{f}_i = h_i^n, \tag{7}
\]

where \(n\) is the total iteration number and \(h_i^n\) is the intermediate feature at \(t^{th}\) iteration. The Eq. 7 can be easily implemented in our SFEM. Specifically, we apply a \(1 \times 1\) convolutional layer to pass the complementary information from \(f_j\) to \(f_i\). \(w_j^t\) is the learned parameter of the convolutional layer and it is shared for all iterations.

As shown in Fig. 3, we apply the proposed SFEM to mutually refine the features in \(\{f^1_2, f^1_2\}, \{f^2_3, f^2_2, f^1_2\}, \{f^3_4, f^3_3, f^2_2, f^1_2\}, \{f^3_4, f^3_3, f^2_2\}\). After receiving the information from other features at different scales, the refined feature becomes more robust to the scale variation of people. The experiments in Section 4 show that our SFEM greatly improves the performance of crowd counting.

### 3.3. Dilated Multiscale Structural Similarity Loss

In this subsection, we employ a Dilated Multiscale Structural Similarity (DMS-SSIM) loss to train our network. We ameliorate the original MS-SSIM [39] with dilation operations to enlarge the diversity of the sizes of local regions and force our network to capture the local correlation within regions of different sizes. Specifically, for each pixel, our DMS-SSIM loss is computed by measuring the structural similarity between the multiscale regions centered at the given pixel on an estimated density map and the corresponding regions on the GT density map.

As shown in Fig. 4, we implement the DMS-SSIM loss
with a dilated convolutional neural network named as DMS-SSIM network. The DMS-SSIM network consists of \( m \) dilated convolutional layers and the parameters of these layers are fixed as a normalized Gaussian kernel with a size of \( 5 \times 5 \) and a standard deviation of \( 1.0 \). The Gaussian kernel is denoted as \( w = \{w(o)\} o \in O, O = \{-2, -2, \ldots, (2, 2)\} \), where \( o \) is an offset from the center.

For convenience, the estimated density map \( M_0 \) in DSSINet is re-marked as \( X_0 \) in this subsection and its corresponding ground-truth is denoted as \( Y_0 \). We feed \( X_0 \) and \( Y_0 \) into the DMS-SSIM network respectively and their outputs at \( i^{th} \) layer are represented as \( X_i \in R^{H \times W} \) and \( Y_i \in R^{H \times W} \). Specifically, for a given location \( p \), \( X_{i+1}(p) \) is calculated by:

\[
X_{i+1}(p) = \sum_{o \in O} w(o) \cdot X_i(p + r_{i+1} \cdot o), \tag{8}
\]

where \( r_{i+1} \) is the dilation rate of the \((i + 1)^{th}\) layer and it is used to control the size of receptive field. Since \( \sum_{o \in O} w(o) = 1 \), \( X_{i+1}(p) \) is the weighted mean of a local region centered at the location \( p \) on \( X_i \) and we could get \( X_{1+1} = \mu_{X_1} \).

In this case, \( X_1 \) is the local mean of \( X_0 \) and \( X_2 \) is the local mean of \( X_1 \). By analogy, \( X_{i+1}(p) \) can be considered as the mean of a relatively large region on \( X_0 \). Based on the filtered map \( X_{i+1} \), we calculate the local variance \( \sigma^2_X(p) \) as:

\[
\sigma^2_X(p) = \sum_{o \in O} w(o) \cdot [X_i(p + r_{i+1} \cdot o) - \mu_{X_i}(p)]^2. \tag{9}
\]

The local mean \( \mu_{Y_i} \) and variance \( \sigma^2_Y \) of filtered map \( Y_i \) are also calculated with the same formulations as Eq. (5) and Eq. (9). Moreover, the local covariance \( \sigma_{X,Y}(p) \) between \( X_i \) and \( Y_i \) can be computed by:

\[
\sigma_{X,Y}(p) = \sum_{o \in O} w(o) \cdot [X_i(p + r_{i+1} \cdot o) - \mu_{X_i}(p)] \cdot [Y_i(p + r_{i+1} \cdot o) - \mu_{Y_i}(p)] \tag{10}
\]

Further, the luminance comparison \( L_i \), contrast comparison \( C_i \) and structure comparison \( S_i \) between \( X_i \) and \( Y_i \) are formulated as:

\[
L_i = \frac{\mu_{X_i} \cdot \mu_{Y_i} + c_1}{\sigma^2_{X_i} + \sigma^2_{Y_i} + c_1}, \quad C_i = \frac{2\sigma_{X,Y} + c_2}{\sigma^2_{X_i} + \sigma^2_{Y_i} + c_2}, \quad S_i = \frac{\sigma_{X,Y} + c_3}{\sigma_{X,Y} + c_3}, \tag{11}
\]

where \( c_1, c_2 \) and \( c_3 \) are small constants to avoid division by zero. The SSIM between \( X_i \) and \( Y_i \) is calculated as:

\[
SSIM(X_i, Y_i) = L_i \cdot C_i \cdot S_i. \tag{12}
\]

Finally, the DMS-SSIM between the estimated density map \( X_0 \) and ground-truth \( Y_0 \), as well as the DMS-SSIM loss are defined as:

\[
DMS-SSIM(X_0, Y_0) = \prod_{i=0}^{m-1}\{SSIM(X_i, Y_i)\}^{\alpha_i}; \tag{13}
\]

\[
Loss(X_0, Y_0) = 1 - DMS-SSIM(X_0, Y_0)
\]

where \( \alpha_i \) is the importance weight of \( SSIM(X_i, Y_i) \) and we refer to [39] to set the value of \( \alpha_i \).

In this study, the number of layers in DMS-SSIM network \( m \) is set to 5. The dilation rates of these layers are 1, 2, 3, 6 and 9 respectively and we show the receptive field of each layer in Table 1. The receptive field of the \( 2^{nd} \) layer is 13, thus \( SSIM(X_1, Y_1) \) indicates the local similarity measured in regions with size of \( 13 \times 13 \). Similarly, \( SSIM(X_4, Y_4) \) is the local similarity measured in \( 85 \times 85 \) regions. When removing the dilation, the receptive fields of all layers are clustered into a small region, which makes DMS-SSIM network fail to capture the local correlation in regions with large heads.

### 4. Experiments

#### 4.1. Implementation Details

**Ground-Truth Density Maps Generation:** In this work, we generate ground-truth density maps with the geometry-adaptive Gaussian kernels [47]. For each head annotation \( p_k \), we mark it as a Gaussian kernel \( \mathcal{N}(p_k, \sigma^2) \) on the density map, where the spread parameter \( \sigma \) is equal to 30% of the mean distance to its three nearest neighbors. Moreover, the kernel is truncated within 3\( \sigma \) and normalized to an integral of 1. Thus, the integral of the whole density map is equal to the crowd count in the image.

**Networks Optimization:** Our framework is implemented with PyTorch [26]. We use the first ten convolutional layers of the pre-trained VGG-16 to initialize the corresponding convolution layers in our framework. The rest of convolutional layers are initialized by a Gaussian distribution with zero mean and standard deviation of \( 1e^{-6} \). At each training
iteration, 16 image patches with a size of $224 \times 224$ are randomly cropped from images and fed into DSSINet. We optimize our network with Adam [14] and a learning rate of $1e^{-5}$ by minimizing the DMS-SSIM loss.

### 4.2. Evaluation Metric

For crowd counting, *Mean Absolute Error* (MAE) and *Mean Squared Error* (MSE) are two metrics widely adopted to evaluate the performance. They are defined as follows:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\hat{P}_i - P_i|,$$

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\hat{P}_i - P_i)^2,$$  \hspace{1cm} (14)

where $N$ is the total number of the testing images, $\hat{P}_i$ and $P_i$ are the estimated count and the ground truth count of the $i^{th}$ image respectively. Specifically, $\hat{P}_i$ is calculated by the integration of the estimated density map.

### 4.3. Comparison with the State of the Art

**Comparison on Shanghaitech [47]:** As the most representative benchmark of crowd counting, Shanghaitech dataset contains 1,198 images with a total of 330 thousand annotated people. This dataset can be further divided into two parts: Part A with 482 images randomly collected from the Internet and Part B with 716 images taken from a busy shopping street in Shanghai, China.

We compare the proposed method with ten state-of-the-art methods on this dataset. As shown in Table 2, our method achieves superior performance on both parts of the Shanghaitech dataset. Specifically, on Part A, our method achieves a relative improvement of 9.5% in MAE and 8.1% in MSE over the existing best algorithm SANet [4]. Although previous methods have worked well on Part B, our method still achieves considerable performance gain by decreasing the MSE from 13.6 to 10.34. The visualization results in Fig. 5 show that our method can generate high-quality crowd density maps with accurate counts, even though the scales of people vary greatly in images.

**Comparison on UCF-QNRF [12]:** The recently released UCF-QNRF dataset is a challenging benchmark for dense crowd counting. It consists of 1,535 unconstrained crowded images (1,201 for training and 334 for testing) with huge scale, density and viewpoint variations. 1.25 million persons are annotated and they are unevenly dispersed to images, varying from 49 to 12,865 per image.

**Comparison on UCF_CC_50 [11]:** This is an extremely challenging dataset. It contains 50 crowded images of various perspective distortions. Moreover, the number of people varies greatly, ranging from 94 to 4,543. Following the standard protocol in [11], we divide the dataset into five parts randomly and perform five-fold cross-validation. We compare our DSSINet with thirteen state-of-the-art methods and we can observe that our DSSINet exhibits the lowest MAE 99.1 and MSE 159.2 on this dataset, outperforming other methods with a large margin. Specifically, our method achieves a significant improvement of 24.9% in MAE over the existing best-performing algorithm CL.

**Comparison on WorldExpo’10 [43]:** As a large-scale crowd counting benchmark with the largest amount of images, WorldExpo’10 contains 1,132 video sequences cap-

### Table 2. Performance comparison on Shanghaitech dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Part A</th>
<th>Part B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MSE</td>
</tr>
<tr>
<td>MCNN [47]</td>
<td>110.2</td>
<td>173.2</td>
</tr>
<tr>
<td>SwitchCNN</td>
<td>90.4</td>
<td>135</td>
</tr>
<tr>
<td>CP-CNN [35]</td>
<td>73.6</td>
<td>106.4</td>
</tr>
<tr>
<td>DNCL [32]</td>
<td>73.5</td>
<td>112.3</td>
</tr>
<tr>
<td>IC-CNN [28]</td>
<td>68.5</td>
<td>116.2</td>
</tr>
<tr>
<td>CSRNet [17]</td>
<td>68.2</td>
<td>115.0</td>
</tr>
<tr>
<td>SANet [4]</td>
<td>67.0</td>
<td>104.5</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>60.63</strong></td>
<td><strong>96.04</strong></td>
</tr>
</tbody>
</table>

### Table 3. Performance of different methods on UCF-QNRF dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>MAE</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCNN [47]</td>
<td>377.6</td>
<td>509.1</td>
</tr>
<tr>
<td>SwitchCNN</td>
<td>318.1</td>
<td>439.2</td>
</tr>
<tr>
<td>CP-CNN [35]</td>
<td>295.8</td>
<td>320.9</td>
</tr>
<tr>
<td>IG-CNN [11]</td>
<td>291.4</td>
<td>349.4</td>
</tr>
<tr>
<td>ConvLSTM [40]</td>
<td>284.5</td>
<td><strong>297.1</strong></td>
</tr>
<tr>
<td>CSRNet [17]</td>
<td>266.1</td>
<td>397.5</td>
</tr>
<tr>
<td>IC-CNN [28]</td>
<td>260.9</td>
<td>365.5</td>
</tr>
<tr>
<td>SANet [4]</td>
<td><strong>258.4</strong></td>
<td>334.9</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>216.9</strong></td>
<td><strong>302.4</strong></td>
</tr>
</tbody>
</table>

### Table 4. Performance comparison on UCF_CC_50.

The results of top two performance are highlighted in red and blue respectively.
To validate the effectiveness of SFEM, we implement the following variants of our DSSINet:

- **W/O FeatRefine**: This model feeds the image pyramid \{I_1, I_2, I_3\} into the three subnetworks, but it doesn’t conduct feature refinement. It takes the original features to generate side output density maps. For example, \(M_1\) is directly generated from \(\hat{f}^{1}_{1.3}, \hat{f}^{1}_{2.2}\) and \(\hat{f}^{1}_{3.2}\).

- **ConcatConv FeatRefine**: This model also takes \{I_1, I_2, I_3\} as input and it attempts to refine multiscale features with concatenation and convolution. For instance, it feeds the concatenation of \(\hat{f}^{1}_{1.3}, \hat{f}^{2}_{2.2}\), \(\hat{f}^{3}_{3.2}\) into a \(1 \times 1\) convolutional layer to compute the feature \(\hat{f}^{3}_{1.3}\), \(\hat{f}^{2}_{2.2}\) and \(\hat{f}^{1}_{3.2}\) are obtained in the same manner.

- **CRF-in FeatRefine**: This model uses the proposed CRFs-based SFEM to refine multiscale features from \{I_1, I_2, I_3\}. We explore the influence of the iteration number \(n\) in CRF, e.g., \(n=1,2,3\).

We train and evaluate all aforementioned variants on Shanghaitech Part A. As shown in Table 6 the variant “W/O FeatRefine” obtains the worst performance for the lack of features refinement. Although “ConcatConv FeatRefine” can reduce the count error to some extent by simply refining multiple features, its performance is still barely satisfactory. In contrast, our SFEM fully exploits the complementarity among multiscale features and mutually refines them with CRFs, and thus significantly boosts the performance. Specifically, our “CRF-2 FeatRefine” achieves the best performance. However, too longer iteration number of CRFs in SFEM would degrade the performance (See “CRF-3 FeatRefine”), since those multiscale features may be excessively mixed and lose their own semantic meanings. Thus, the iteration number \(n\) in CRFs is 2 in our final model.

**Influence of the Scale Number of Image Pyramid**: To validate the effectiveness of multiscale input, we train mul-

---

**Figure 5.** Visualization of the crowd density maps generated by our method on Shanghaitech Part A. The first row shows the testing images with people of various scales. The second row shows the ground-truth density maps and the standard counts, while the third row presents our generated density maps and estimated counts. Our method can generate high-quality crowd density maps with accurate counts, even though the scales of people vary greatly in images.

<table>
<thead>
<tr>
<th>Method</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang et al</td>
<td>9.8</td>
<td>14.1</td>
<td>14.3</td>
<td>22.2</td>
<td>3.7</td>
<td>12.9</td>
</tr>
<tr>
<td>MCNN [47]</td>
<td>3.4</td>
<td>20.6</td>
<td>12.9</td>
<td>13.0</td>
<td>8.1</td>
<td>11.6</td>
</tr>
<tr>
<td>ConvLSTM [40]</td>
<td>7.1</td>
<td>15.2</td>
<td>15.2</td>
<td>13.9</td>
<td>3.5</td>
<td>10.9</td>
</tr>
<tr>
<td>SwitchCNN [30]</td>
<td>4.4</td>
<td>15.7</td>
<td>10.0</td>
<td>11.0</td>
<td>5.9</td>
<td>9.4</td>
</tr>
<tr>
<td>DNCL [52]</td>
<td>1.9</td>
<td>12.1</td>
<td>20.7</td>
<td>8.3</td>
<td>2.6</td>
<td>9.1</td>
</tr>
<tr>
<td>CP-CNN [35]</td>
<td>2.9</td>
<td>14.7</td>
<td>10.5</td>
<td>10.4</td>
<td>5.8</td>
<td>8.86</td>
</tr>
<tr>
<td>CSRNet [17]</td>
<td>2.9</td>
<td>11.5</td>
<td>8.6</td>
<td>16.6</td>
<td>3.4</td>
<td>8.6</td>
</tr>
<tr>
<td>SANet [4]</td>
<td>2.6</td>
<td>13.2</td>
<td>9.0</td>
<td>13.3</td>
<td>3.0</td>
<td>8.2</td>
</tr>
<tr>
<td>DRSAN [20]</td>
<td>2.6</td>
<td>11.8</td>
<td>10.3</td>
<td>10.4</td>
<td>3.7</td>
<td>7.76</td>
</tr>
<tr>
<td>ACSCP [31]</td>
<td>2.8</td>
<td>14.05</td>
<td>9.6</td>
<td>8.1</td>
<td>2.9</td>
<td>7.5</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td>1.57</td>
<td>9.51</td>
<td>9.46</td>
<td>10.35</td>
<td>2.49</td>
<td>6.67</td>
</tr>
</tbody>
</table>

Table 5. MAE of different methods on the WorldExpo’10 dataset.
We also discuss the complexity of our method. As the subnetworks in our framework have shared parameters and the kernel size of the convolutional layers in SFEM is $1 \times 1$, the proposed DSSINet only has 8.858 million parameters, 86.19% (7.635 million) of which come from its backbone network (the first ten convolutional layers of VGG-16). As listed in Table 9, the number of parameters of our DSSINet is only half of that of the existing state-of-the-arts (e.g. CSRNet). Compared with these methods, our DSSINet achieves better performance with much fewer parameters. During the testing phase, DSSINet takes 450 ms to process a 720×576 frame from SD surveillance videos on an NVIDIA 1080 GPU. This runtime speed is already qualified for the needs of many practical applications, since people do not move so fast and not every frame needs to be analyzed.

5. Conclusion

In this paper, we develop a Deep Structured Scale Integration Network for crowd counting, which handles the huge variation of people’s scales from two aspects, including structured feature representation learning and hierarchically structured loss function optimization. First, a Structured Feature Enhancement Module based on conditional random fields (CRFs) is proposed to mutually refine multiple features and boost their robustness. Second, we utilize a Dilated Multiscale Structural Similarity Loss to force our network to learn the local correlation within regions of various sizes, thereby producing locally consistent estimation results. Extensive experiments on four benchmarks show that our method achieves superior performance in comparison to the state-of-the-art methods.
References


