Uniformity in Heterogeneity: Diving Deep into Count Interval Partition for Crowd Counting

Changan Wang[1]* Qingyu Song[1]* Boshen Zhang[1] Yabiao Wang[1]
Ying Tai[1] Xuyi Hu[1,3] Chengjie Wang[1]† Jilin Li[1] Jiayi Ma[4] Yang Wu[2]
[1]Tencent Youtu Lab, [2]Applied Research Center (ARC), Tencent PCG
[3]Department of Electronic & Electrical Engineering, University College London, United Kingdom
[4]Electronic Information School, Wuhan University, Wuhan, China
{changanwang, boshenzhang, caseywang, yingtai, jasoncjwang, jerolinli}@tencent.com,
qingyusong@zju.edu.cn, zceexhu@ucl.ac.uk, jyma2010@gmail.com, dylanywu@tencent.com

Abstract

Recently, the problem of inaccurate learning targets in crowd counting draws increasing attention. Inspired by a few pioneering work, we solve this problem by trying to predict the indices of pre-defined interval bins of counts instead of the count values themselves. However, an inappropriate interval setting might make the count error contributions from different intervals extremely imbalanced, leading to inferior counting performance. Therefore, we propose a novel count interval partition criterion called Uniform Error Partition (UEP), which always keeps the expected counting error contributions equal for all intervals to minimize the prediction risk. Then to mitigate the inevitably introduced discretization errors in the count quantization process, we propose another criterion called Mean Count Proxies (MCP). The MCP criterion selects the best count proxy for each interval to represent its count value during inference, making the overall expected discretization error of an image nearly negligible. As far as we are aware, this work is the first to delve into such a classification task and ends up with a promising solution for count interval partition. Following the above two theoretically demonstrated criterions, we propose a simple yet effective model termed Uniform Error Partition Network (UEPNet), which achieves state-of-the-art performance on several challenging datasets. The codes will be available at: TencentYoutuResearch/CrowdCounting-UEPNet.

1. Introduction

The task of crowd counting estimates the number of people in given images or videos. It has drawn remarkable attention in recent years, since its wide range of practical applications in public safety. Currently, most existing state-of-the-art methods adopt density maps [1, 18, 19] as the targets and learn with Convolutional Neural Networks (CNNs).

However, the widely used learning targets, *i.e.*, the density maps, are actually inaccurate and tend to be noisy, as discussed in [1, 12, 17, 20, 23]. These imperfect density maps are caused by several factors, such as empirically selected Gaussian kernels, large density variations and label-

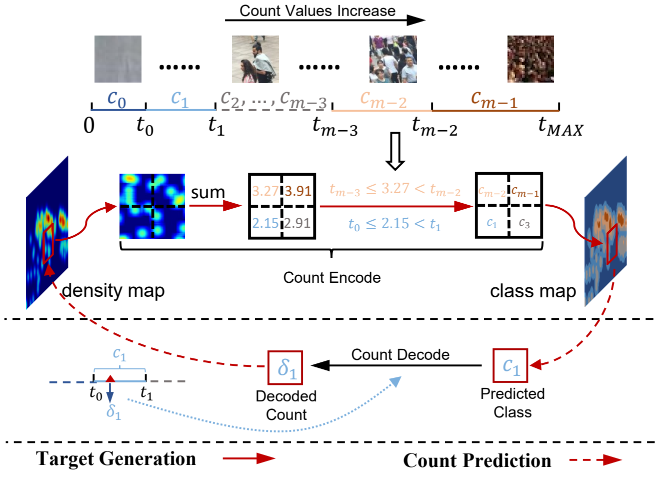


Figure 1: The illustration of our counting pipeline. For image patches with count values between t_i and t_{i+1}, our UEPNet learns to classify them into one (*i.e.*, c_{i+1}) of the m intervals. The count interval borders $(t_0, ..., t_{m-2})$ are determined by the proposed UEP criterion to ensure the minimum expected prediction error. We assign the ground truth class of a patch by the count interval it falling into. During inference, for a patch classified to the interval class of c_i, we set its predicted count to be the count proxy value σ_i of c_i. And the count proxy is a specific value selected from that interval with our proposed MCP criterion.

*Equal contribution. †Corresponding author.

ing deviations. As shown in the supplementary materials, the inaccurate density map introduces severe *inconsistency* between the semantic content and the ground truth target, acting as a kind of "noises" in the learning target. Therefore, when learning the exact count values with an outlier-sensitive loss, such as the commonly used Mean Squared Error (MSE), the model may tend to overfit these inaccurate and ambiguous "noises", leading to inferior performance and poor generalization ability [12].

Inspired by a few pioneering works [3, 12, 21, 25], we solve the above problem with the paradigm of local count (the count value of a local patch) classification, as shown in Figure 1. In such a paradigm, the local count is quantified to one of several pre-defined count interval bins, depending on the interval it falls into. Then we resorts to count interval classification by taking each interval as a single class. However, the number of samples with increasing local counts follows a long-tailed distribution, leading to imbalanced training data and extreme counting errors for some count intervals. What makes the problem worser is that samples from different intervals have various learning difficulty, thus also contribute different counting errors. In this paper, firstly, we solve the above problem from the perspective of designing an optimal interval partition strategy. Specifically, we derive an Uniform Error Partition (UEP) criterion following *the principle of maximum entropy*. Without any prior over the density distribution of an unseen image, the UEP criterion keeps the expected counting errors nearly equal for all intervals to minimize the prediction risk. Such a novel strategy provides a comprehensive consideration of misclassification errors and sample imbalance problem.

Secondly, another crucial yet unsolved problem of this paradigm raises from inference stage. Specifically, a fixed value (*i.e.*, count proxy) should be selected from each count interval, using as the predicted counts for those patches classified to this interval. Obviously, mapping a range of local counts to a single count value would inevitably introduce perceivable discretization errors. To mitigate the above problem, we propose a Mean Count Proxies (MCP) criterion for the count proxy selection. The MCP criterion uses average count value of samples in an interval as the optimal count proxy, with which the expected discretization error is theoretically demonstrated to be nearly negligible.

Thirdly, since the distribution of count values is essentially continuous, it is tricky to unambiguously classify samples whose count values are located around the interval borders. Thus, we design two parallel prediction heads with overlapping count intervals, named Interleaved Prediction Heads (IPH). In this reciprocal prediction structure, the samples whose count values fall near the interval borders of one head are more likely to be correctly classified in another head, reducing the misclassification errors due to the learning ambiguity around interval borders. We conduct extensive experiments on several public datasets with various crowd densities to show the consistent improvements. The contributions of this paper are summarized as follows:

1. We propose a novel criterion termed UEP to tackle with the crucial yet unsolved count interval partition problem, making the local count classification based counting models achieve the minimum prediction risk.

2. We propose the MCP criterion to minimize the discretization errors inevitably introduced by the count quantification, which is orthogonal to the UEP criterion.

3. We propose a simple yet effective model termed UEP-Net, following the above two key criterions. Combining with the IPH, the UEPNet achieves state-of-the-art performance on several benchmarks.

2. Related Work

In this section, we review the recent density map learning based methods in the literature. Especially, we reveal the defects in very few existing local count classification based methods to show the necessity of our research. We also discuss some works dealing with the imperfect targets to show the superiority of our method.

Pixel-wise Density Maps Learning. The concept of density maps is firstly introduced by Lempitsky and Zisserman [10], and since then this kind of learning targets has been widely used in subsequent methods [2, 8, 11, 28]. These methods are trained with pixel-wise supervision and finally obtain the estimated count by summing over the predicted density maps. Although these methods [5, 9] achieved good performance, they still suffer from the following defects. Firstly, they ignore the negative impacts of the imperfect density maps and learn to overfit these inaccurate targets with an outlier-sensitive loss. Secondly, there exists inconsistency between the minimum training loss and the optimal counting result. These problems limit the counting accuracy and generalization ability of such approaches.

Patch-wise Density Maps Learning. Instead of pixel-wise regression, patch-wise density maps learning based methods predict the count values of local patches. This idea is firstly used in S-DCNet [25], and is helpful to alleviate the inconsistency between training target and evaluation criteria as demonstrated in [16]. Among these methods, some works [24] adopt regression with supervision from the commonly used MSE loss but still suffer from the inaccurate targets. Thus the other methods [12, 25] discretize the local counts and learn to classify count intervals. However, despite the robustness of such a paradigm, both of them divide the count intervals intuitively, which is inappropriate and greatly limits the performance. Besides, the two methods directly use median value of the interval to represent its class count, which is sub-optimal and introduces additional discretization errors. We also adopt the paradigm of patch-

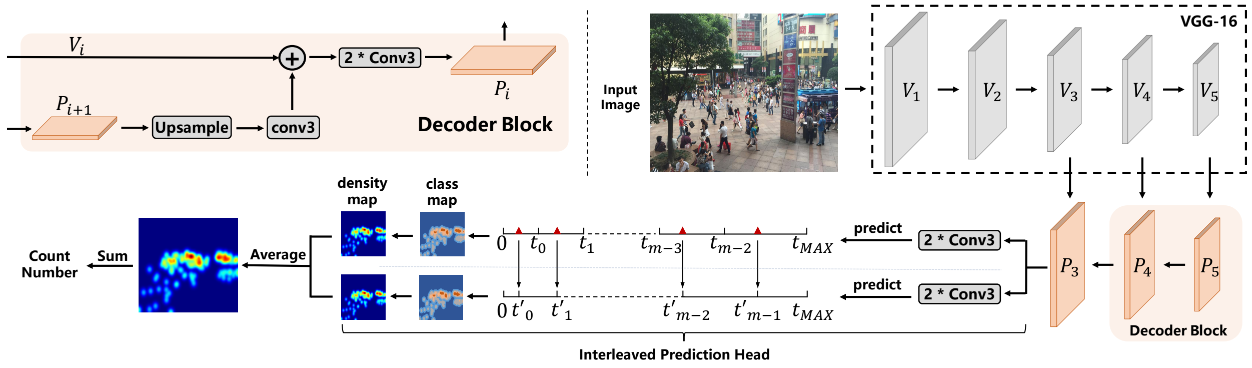


Figure 2: The network structure of the proposed UEPNet. It is a fully convolutional model, and consists of a simple encoder-decoder network for feature extraction and an Interleaved Prediction Head to classify each patch into certain interval. There are two parallel heads with overlapping count intervals to predict the classes independently, and their decoded counts are averaged to get final result. The ground truth class map is encoded from density map, guided by the proposed UEP criterion. The predicted class can be decoded back to local count, in which the count proxy (as denoted by the Red triangle) is selected with the proposed MCP criterion. The UEPNet is optimized with the Cross Entropy loss in an end-to-end manner.

wise count classification, but focus more on its crucial yet unsolved count interval partition problem.

Learning with Inaccurate Targets. Recently, there has been increasing attention on the problem of inaccurate ground truth targets. One solution is to improve the quality of the generated density maps by estimating more accurate Gaussian kernel [20, 23], while the improvement seems to be limited by insufficient annotations. Bayesian loss is proposed by [17] to provide a more reliable supervision on the count expectation at each annotated point, but it still adopts a pixel-wise regression. Moreover, ADSCNet [1] proposes a self-correction supervision to iteratively correct the annotations with the model estimations, which however ignores the additional noises introduced by those inaccurate estimations during the online updating. Differently, we resort to local count classification, which is more robust than learning the exact but inaccurate count values [12].

3. Our Approach

We firstly introduce the symbols used in this paper in Sec. 3.1. After that, we introduce the pipeline of our framework in detail. Specifically, we begin with the calculation of the local counts in an image as described in Sec. 3.2. Then we exhibit our strategy of quantifying these local counts into different intervals in Sec. 3.3. After an end-to-end training supervised with Cross Entropy, we obtain the predicted count of an image patch by the MCP criterion, as described in Sec. 3.4. Finally, in Sec. 3.5, the novel prediction module IPH is described in detail.

3.1. Symbol Definition

Table 1 lists the main symbols used in the following text. In this paper, we conduct a patch-level analysis by breaking down the whole training set into K image patches with a same size. For the k-th image patch, the summation of its density values is calculated as its local count d_k. Then

Symbol	Definition
$\mathcal{E}$	The expected counting error.
$\tilde{\mathcal{E}}$	The absolute mean counting difference.
$\mathcal{T}$	All local counts in the training set.
$d_k, k \in \{1, 2, ..., K\}$	k-th local count in $\mathcal{T}$.
m	Total number of intervals.
$t_0, ..., t_{m-2}$	Count values at interval borders.
t_{MAX}	The max local count in $\mathcal{T}$.
$c_i, i \in \{0, 1, ..., m-1\}$	i-th interval.
x_{ij}	j-th local count from c_i.
n_i	Number of samples in c_i.
l_i	Length of c_i.
δ_i	Count proxy of c_i.

Table 1: Main symbols definition.

there should be a collection $\mathcal{T}$ containing all the local counts of the above K samples. For an arbitrary partition setting with m intervals, we denotes the i-th interval as c_i. And the length l_i of c_i is determined by its left border t_i and right border t_{i+1}. In other words, the count interval c_i is a contiguous count value range from t_i to t_{i+1}. Then for any sample x_{ij} ($j \in \{1, ..., n_i\}$) whose local count falls into the interval c_i, we assign a fixed count value δ_i (termed as count proxy) as its predicted count if it is correctly classified. $\mathcal{E}$ is the expectation of the counting error for a randomly given unseen image. $\tilde{\mathcal{E}}$ is the absolute value of the average counting difference from multiple images or patches, in which the counting difference is calculated by the subtraction of ground truth count and predicted count.

3.2. Local Count Calculation

For an image containing n heads, the ground truth annotations can be expressed as a dot map $M = \sum_{i=1}^{n} \delta(p - p_i)$, where p_i is the point annotation for the i-th head and the delta function $\delta(p - p_i)$ represents there is one head at pixel p_i. Then we generate density map D by convolving over M using a Gaussian kernel G_σ, which can be described as

$D = \sum_{i=1}^{n} \delta(p - p_i) * G_\sigma$. The spread parameter σ in G_σ is fixed or determined by the k nearest neighbors [28]. With a normalized Gaussian kernel, the counting results can be obtained by summing over the density maps.

Based on the density map D, we can construct a local count map D_s, in which each value is the head count from the corresponding patch of size $s \times s$ in the image. Specifically, we slide a window of size $s \times s$ without overlapping over the density map D and sum the density values in this window as the local count d.

3.3. The Interval Partition Strategy

We derive our key partition strategy from the perspective of minimizing the expected counting error $\mathcal{E}$ for an unseen image. And $\mathcal{E}$ is obtained by calculating the absolute value of the difference between the estimated count and the ground truth count at the image level. Since we have no prior over the density distribution of any given unseen image, we propose to approximate $\mathcal{E}$ from a perspective of probability sampling. Specifically, we firstly decompose $\mathcal{E}$ as the absolute value of the summation for the counting differences from multiple non-overlapping patches of that image. Then under the assumption of independent and identically distributed (i.i.d.) data, these images patches are seen as a random sampling subset from K image patches in the training set. Thus, minimizing $\mathcal{E}$ for a randomly given image is equivalent to achieving the minimum $\tilde{\mathcal{E}}$ on all patches in the sampled subset. The above approximation helps us to design a generalized partition strategy using the available training set, as demonstrated in Supplementary Materials.

Since our goal is to minimize $\tilde{\mathcal{E}}$ with a proper count interval partition strategy, we would like to figure out how the counting errors from different intervals contribute to $\tilde{\mathcal{E}}$. However, for any given unseen image, the patches in its equivalent subset are always randomly selected. In other words, for a specific image, we have no way to determine the patch distribution over all intervals, neither its counting error contribution from different intervals. In such a case, according to *the principle of maximum entropy*, we should not make further assumptions to this distribution. As a result, we propose an Uniform Error Partition (UEP) criterion, which always keeps the expected counting error equal for all intervals. With the proposed UEP criterion, the risk of making extreme prediction error is minimized, achieving reasonable $\tilde{\mathcal{E}}$ for any given image. From another perspective, we could achieve a minimum $\tilde{\mathcal{E}}$ in terms of mathematical expectation with the UEP criterion.

Before conducting the above UEP criterion, a key step is to estimate the counting error from a certain interval. In this part, we estimate the interval-level count error based on all samples in $\mathcal{T}$, because every element within it has the same chance to be selected into the equivalent subset for an image. For a specific interval c_i, its counting error contribution depends on both the sample number n_i within it and the misclassification cost of each sample from it. The former one is proportional to the interval length l_i, which is relatively easier to obtain from the dataset statistics. Besides, for a single sample x_{ij} from interval c_i, it is more likely to be misclassified to a nearby interval. Since the interval lengths of adjacent intervals are nearly equal, the misclassification cost is also proportional to the interval length l_i. Therefore, we reasonably take $n_i l_i$ as the approximation for the counting error from c_i. By keeping a uniform $n_i l_i$ for all intervals, the UEP criterion takes a fully consideration on the misclassification cost and the sample imbalance problem among intervals, making the classification task more easier to learn. We provide more detailed analysis in the Supplementary Materials.

Algorithm 1: Binary Interval Partition with UEP

```
Input: $\mathcal{T} \leftarrow$ local count collection in the training set
       $m \leftarrow$ required number of intervals
       $[L, H] \leftarrow$ search range for $n_i l_i$
Output: $\mathcal{P} \leftarrow$ interval endpoint collection
/* remove counts less than $t_0$ */
1  $\mathcal{T} \leftarrow sort(filter(\mathcal{T}))$
2  while $abs(H - L) > \epsilon$ do
       /* try to divide by $n_i l_i = \bar{l}$ */
3      $p \leftarrow t_0, n \leftarrow 0, \mathcal{P} \leftarrow \{t_0\}, \bar{l} \leftarrow (L + H)/2$
4      for $d_k$ in $\mathcal{T}$ do
5          $n \leftarrow n + 1$
           /* an endpoint is found */
6          if $(d_k - p) \times n > \bar{l}$ then
7              $n \leftarrow 0, p \leftarrow d_k, \mathcal{P} \leftarrow \mathcal{P} \cup \{p\}$
       /* adjust $[L, H]$ by $|\mathcal{P}|$ and $m$ */
8      if $|\mathcal{P}| >= m$ then
9          $L \leftarrow \bar{l}$                // when $|\mathcal{P}| >= m$
10     else
11         if $|\mathcal{P}| == m - 1$ then
               /* adjust by $n_{m-1} l_{m-1}$ */
12             if $(t_{MAX} - p) \times n > \bar{l}$ then
13                 $L \leftarrow \bar{l}$
14             else
15                 $H \leftarrow \bar{l}$
16         else
17             $H \leftarrow \bar{l}$            // when $|\mathcal{P}| < m - 1$
18 return $sort(\mathcal{P})$
```

To apply the UEP criterion during the interval partition process, we propose a Binary Interval Partition (BIP) algorithm, as shown in Algorithm 1. For a required interval number m, the BIP algorithm determines a target $\hat{nl}$ for all intervals by an iterative binary searching process. By

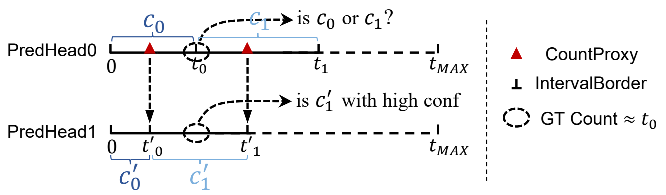


Figure 3: The interval partition setting in the proposed IPH. The interval borders in PredHead0 are determined by the UEP criterion. While the interval borders in PredHead1 are automatically obtained by directly using the count proxy values in PredHead0, which helps to mitigate the classification ambiguity around the interval borders in PreadHead0.

adjusting $n_i l_i$ to be equal with $\hat{nl}$, the whole count range from 0 to t_{MAX} could be equally divided into m intervals. Specifically, we obtain the interval borders by a single pass along the local count histogram of $\mathcal{T}$. During the pass, we keep pushing the right boundary t_{i+1} of an interval c_i until $n_i l_i \approx \hat{nl}$. One in particular is the count interval $[0, t_0)$ for the background class, we manually set t_0 due to the extreme number of background samples.

3.4. The Count Proxy Selection Strategy

Intuitively, assigning a fixed count value for all patches classified to an interval inevitably introduces discretization errors. This problem becomes even more perceivable when the number of intervals is relatively small. However, increasing the number of intervals would lead to heavier computational burden in the classification head. Instead, to minimize the discretization errors, we propose a count proxy selection criterion termed as Mean Count Proxies (MCP). The MCP criterion demonstrates that the expected discretization error could be nearly negligible, as long as we choose the average count value of samples in corresponding interval as its count proxy.

Actually, when all the patches in an image are classified correctly, its expected counting error $\mathcal{E}$ degenerates to the discretization error. The above observation provides us a way to quantify the discretization error. Taking the interval c_i as an example, all the correctly classified samples x_{ij} ($j \in \{1, ..., n_i\}$) within c_i shares a same count proxy δ_i. So we could further conclude that if we let $\delta_i = \sum_{j=1}^{n_i} x_{ij}/n_i$, $\mathcal{E}$ will get the minimal value 0. In other words, with the proposed MCP criterion, there will be *no extra quantization errors* when transforming the regression task into an interval classification problem. We provide detailed mathematical proof in the Supplementary Materials.

3.5. The Interleaved Prediction Heads

In this part, we introduce the proposed Interleaved Prediction Heads (IPH). Intuitively, the samples whose count values fall near the boundaries of intervals are particularly difficult to be correctly classified, which is due to the learning ambiguity around the interval borders. These samples tend to contribute more to the misclassification errors, especially for those samples from the intervals with relatively larger length.

To alleviate the above problem, we parallelly duplicate the prediction head and use overlapping count intervals for them. Specifically, as shown in Figure 3, the interval borders in the first head (PredHead0) are obtained using the UEP criterion. While the other head (PredHead1) directly use the count proxies in PredHead0 as its interval borders (except for the count interval $[0, t_0)$ of the background class). The count proxies of PredHead1 are independently calculated using the MCP criterion. The two heads are trained with CE loss independently, and during inference their predicted local count maps are averaged as the final estimation. With the IPH, the samples whose count values fall near the interval borders of one head are more likely to be correctly classified in the other head.

4. Experiments

We conduct extensive experiments on four challenging benchmarks with various crowd densities. Mean Absolute Error (MAE) and Root Mean Squared Error (MSE) are used as the evaluation metrics following [28].

4.1. Implementation Details

Datasets. Experimental evaluations for the UEPNet are conducted on four widely used crowd counting benchmark datasets: ShanghaiTech [28] part A and part B, UCF_CC_50 [6], UCF-QNRF [7], and WorldExpo'10 [27]. The count statistics for these datasets can be found in a recent survey [4]. We generate ground truth density maps for ShanghaiTech part B and WorldExpo'10 using Gaussian kernel G_σ with fixed sigma of 15.0 and 5.0 respectively. For the other datasets we use geometry-adaptive Gaussian kernel following [28]. We follow the standard protocol in [6] to conduct a five-fold cross validation on UCF_CC_50. For WorldExpo'10, we blur image regions outside the provided RoIs using mean filter of kernel size 11.

Data Augmentations. Scaling with a random factor in [0.3, 1.3] and random mirroring are performed to increase the diversity of image patches. Then we random select a cropped patch for training following CSRNet [11]. Since images in UCF-QNRF have extremely large resolution, we limit the crop size within 1024 and make inference with a sliding window of size 1024 due to the limited memory.

Hyperparameters. The size of local patches in UEPNet is 8×8, since we use the feature map of stride 8 for predictions. As demonstrated, the interval number works well in a wide range, and is empirically selected as 25 for all datasets. The ϵ in Algorithm 1 is set to 100, and t_0 is set to $1.6e^{-4}$.

Methods	SHTechPartA		SHTechPartB		UCF_CC_50		UCF-QNRF		WorldExpo'10					
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	S1	S2	S3	S4	S5	avg.
CAN [15]	62.3	100.0	7.8	12.2	212.2	**243.7**	107.0	183.0	2.9	12.0	10.0	7.9	4.3	7.4
CSRNet [11]	68.2	115.0	10.6	16.0	266.1	397.5	-	-	2.9	11.5	8.6	16.6	3.4	8.6
PGCNet [26]	57.0	**86.0**	8.8	13.7	244.6	361.2	-	-	2.5	12.7	8.4	13.7	3.2	8.1
DSSINet [14]	60.63	96.04	6.85	10.34	216.9	302.4	99.1	159.2	1.57	9.51	9.46	10.35	**2.49**	6.67
Liu et al. [12]	62.8	102.0	8.6	16.4	239.6	322.2	141.0	219.0	2.1	11.4	**8.1**	16.8	2.5	8.2
BL+ [17]	62.8	101.8	7.7	12.7	229.3	308.2	88.7	154.8	-	-	-	-	-	-
S-DCNet [25]	58.3	95.0	6.7	10.7	204.2	301.3	104.4	176.1	-	-	-	-	-	-
DUBNet [19]	64.6	106.8	7.7	12.5	243.8	329.3	105.6	180.5	-	-	-	-	-	-
SDANet [18]	63.6	101.8	7.8	**10.2**	227.6	316.4	-	-	2.0	14.3	12.5	9.5	2.5	8.1
ADSCNet [1]	55.4	97.7	6.4	11.3	198.4	267.3	**71.3**	132.5	-	-	-	-	-	-
LibraNet [13]	55.9	97.1	7.3	11.3	181.2	262.2	88.1	143.7	-	-	-	-	-	-
AMSNet [5]	56.7	93.4	6.7	**10.2**	208.4	297.3	101.8	163.2	1.6	**8.8**	10.8	10.4	2.5	6.8
ASNet [9]	57.78	90.13	-	-	174.84	251.63	91.59	159.71	2.22	10.11	8.89	**7.14**	4.84	**6.64**
AMRNet [16]	61.59	98.36	7.02	11.00	184.0	265.8	86.6	152.2	-	-	-	-	-	-
Ours	**54.64**	91.15	**6.38**	10.88	**165.24**	275.90	81.13	**131.68**	**1.56**	9.83	10.31	8.56	4.13	6.88

Table 2: Comparisons with SOTA methods on four datasets. The best performance is bold and the second best is underlined.

The first 13 convolutional layers in UEPNet are initialized from a pre-trained VGG-16 [22] model, and the rest layers are randomly initialized by a Gaussian distribution with the mean of 0 and the standard deviation of 0.01. We fine-tune the model using SGD with batch size 1. The learning rate is initially set to 0.001 and is decreased by 1/10 when the training loss stagnates. We keep training until convergence.

4.2. Comparisons with State-of-the-Art Methods

We compare our UEPNet with state-of-the-art methods on the datasets described above. As shown in Table 2, our UEPNet achieves state-of-the-art performance on these datasets. Compared with the previous strongest method ADSCNet, which achieved the best MAE on several datasets, UEPNet achieves better MAE on SHTech Part B. And the UEPNet surpasses ADSCNet with a 1.4% relative improvement in MAE on SHTech Part A, with a 16.7% relative improvement in MAE on UCF_CC_50. As for UCF-QNRF, most methods give very few details of the testing procedure including ADSCNet. With a simple patch-wise inference (due to the limited GPU memory), our UEPNet achieves higher MAE than ADSCNet, but already reduces the MAE by 5.7% compared with the second-best method AMRNet. Besides, the BL+ method designed a novel loss to deal with the inaccurate targets, while our UEPNet, which is supervised with a simple CE loss, outperforms BL+ by a large margin on all datasets. Generally speaking, although using a quite simple network structure, the UEPNet performs better than previous methods, even for the Neural Architecture Search based method AMSNet.

We further make comparison with the other two methods, Two-Linear in S-DCNet [25] and Liu et al. [12], to show the superiority of our strategy for interval partition and count proxy selection. It is worth mentioning that the two methods also use the paradigm of counting by local count classification, but both of them divide the count intervals intuitively. As shown in Table 2, our UEPNet achieves significant improvement on all the four datasets. For more fair comparison, we firstly remove the IPH and use median value of the interval as its count proxy. Then in order to use their carefully tuned hyperparameters, we conduct experiments under different strides with the same network structure and report the results in Table 3. The consistent and significant improvements indicate the crucial of count interval design and the effectiveness of our UEP criterion.

Metrics	Stride8	Stride16		Stride32	
	Ours	S-DCNet	**Ours**	Liu et al.	**Ours**
MAE	7.72	9.63	7.71	8.81	7.65
MSE	12.51	14.22	12.80	14.02	12.28

Table 3: Comparisons with other interval partition methods on SHTech Part B using the same network structure and the same number of intervals.

4.3. Ablation Studies

MCP	IPH	PPH	MS	SHTech Part A		SHTech Part B	
				MAE	MSE	MAE	MSE
				61.56	102.30	7.72	12.51
✓				58.69	97.64	7.64	13.18
✓			✓	57.97	100.34	6.87	11.33
✓	✓			56.80	92.01	7.25	11.91
✓		✓		58.47	97.51	7.57	12.88
✓	✓		✓	**54.64**	**91.15**	**6.38**	**10.88**

Table 4: Ablation studies of our contributions. MS in the table is short for the multi-scale training. PPH is a variant of the IPH, and the two heads in the PPH use exactly the same interval setting instead of the overlapping intervals.

In this section, we demonstrate the effectiveness of our contributions through ablation studies on two datasets with various density, *i.e.*, SHTech Part A and SHTech Part B.

Effect of UEP. As shown in Table 3, dividing count intervals with our UEP criterion performs better than the previous two intuitively designed interval settings, with 20.0%

Methods	Median		MCP	
	MAE	MSE	MAE	MSE
UniformLen	82.62	90.65	13.14	17.17
UniformNum	62.46	75.26	10.88	25.62
Ours	**7.72**	**12.51**	**7.64**	**13.18**

Table 5: Comparison with other interval settings on SHTech Part B. We use Median to represent using the median value of each interval as its count proxy.

and 13.2% improvements on MAE respectively. To highlight the benefit of UEP criterion, which takes a comprehensive consideration of the misclassification cost and the sample imbalance problem among intervals, we make comparison with another two interval settings, *i.e.* UniformLen and UniformNum. UniformLen represents the strategy of dividing intervals with equal interval length, and UniformNum represents the strategy of dividing intervals with equal number of samples in each interval. From Table 5, our method significantly outperforms UniformLen and UniformNum.

Effect of MCP. As shown in Table 5, compared with the median count proxies, the adoption of our MCP consistently reduces the MAE of UniformLen and UniformNum, with 20.0% and 13.2% reduction respectively. From the ablation studies in Table 4, with the MCP, the MAE improves from 61.56 to 58.69 on Part A and from 7.72 to 7.64 on Part B.

Effect of IPH. From the ablation results in Table 4, the IPH indeed reduces the estimation errors, with 3.2% and 5.1% improvements on MAE for SHTech Part A and SHTech Part B respectively. Due to the augmentation of multi-scale training, the relative improvements for IPH are a little higher (5.7% and 7.1%). To eliminate the gain of voting ensemble, we conduct experiments using Parallel Prediction Heads (PPH), which uses exactly the same interval setting instead of the overlapping intervals for the two heads. As shown in Table 4, the improvement of PPH is only 0.4% for SHTech Part A and 0.9% for SHTech Part B.

Effect of Multi-Scale Training. This data augmentation is particularly helpful to the classification related task, *i.e.*, the local count classification, which significantly increases the diversity of samples in each interval. As shown in Table 4, with multi-scale training, the MAE of our UEPNet reduces from 56.80 to 54.64 on SHTech Part A and from 7.25 to 6.38 on SHTech Part B. However, as demonstrated in the following, the benefit of multi-scale training is only valid for the paradigm of counting by local count classification.

4.4. Discussions

Why classification works better? It is natural to ask that why the task of local count classification could perform better than regression task. Besides the promising counting accuracy as shown in [12, 13, 25], we provide an intuitive explanation. Firstly, it only adopts a Cross Entropy (CE) loss, which is more robust to outliers compared with the commonly used MSE loss in regression. Secondly, it is much easier to identify that if a local count falls into a certain interval than determining its accurate value. Thirdly, the regression task requires to strictly predict the exact counts of two patches, although their ground truth counts might be inaccurate. Differently, for the classification task, only when the gap between their counts is large enough (beyond the length of an interval), the two patches might be classified as different classes. Under the existence of inaccurate targets, this more relaxed supervision helps to stabilize the training process, avoiding overfitting or local optimum. Finally, such a paradigm shows its potential of achieving superior performance when using our proposed criterions.

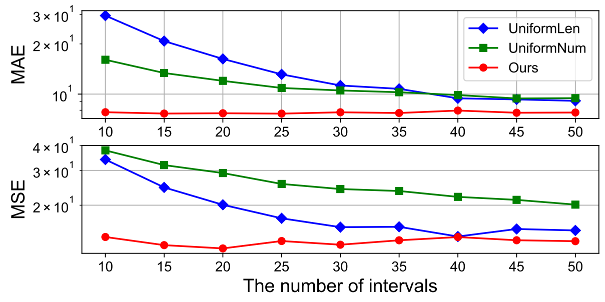


Figure 4: The effect of the number of intervals.

How to determine the class number? We conduct experiments with different number of intervals. As shown in Figure 4, our UEPNet is robust to the number of intervals, while the methods of UniformLen and UniformNum get much higher MAE when using lower number of intervals. As we increase the number of intervals, the MAE of UniformLen and UniformNum starts to reduce and finally stagnates, but is still higher than our method. Consequently, what matters is the interval partition strategy rather than the number of intervals, and our method is insensitive to the number of intervals. We empirically recommend to use 25 classes, which works well on all dataset in our experiments.

	S-DCNet	Liu et al.	Ours
Median	17.84	11.04	1.88
MCP	3.20	1.70	**0.27**

Table 6: The discretization errors on SHTech Part B test set.

The analysis of discretization errors. We quantify the discretization errors by assuming all samples are correctly classified, which eliminates the impact from the misclassification errors. Specifically, for the local count d_k of each patch (without overlapping) in an image from $\mathcal{I}_{test}$, if d_k falls into an interval $l_k, l_k \in \{0, 1, ..., m-1\}$, its predicted count $\hat{d_k}$ should be the count proxy of interval l_k, *i.e.*, $\hat{d_k} = \delta_{l_k}$. Then, we calculate $|\sum_{d_k \in \mathcal{I}_{test}}(d_k - \hat{d_k})|$ as the discretization errors on $\mathcal{I}_{test}$, since no misclassification errors are introduced during this process. As shown in Table 6, with MCP, the average discretization errors of all test images are significantly reduced. Combining with UEP and MCP, the discretization errors are nearly negligible.

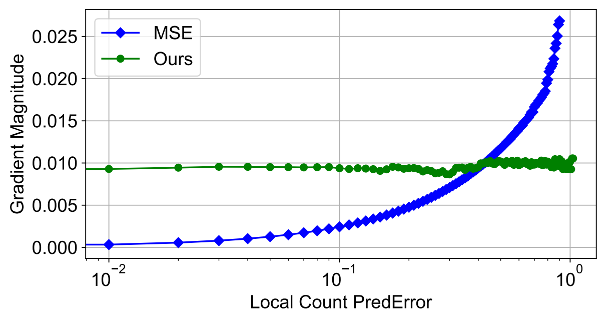


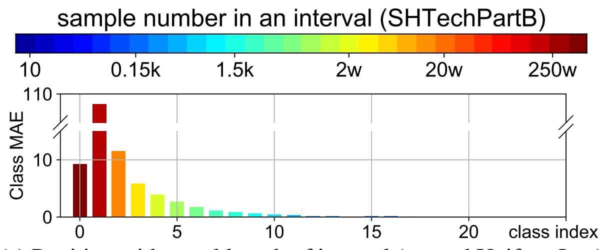
Figure 5: Comparison of gradient contributions in the converged models using MSE and our approach.

The effectiveness of our interval partition strategy. As illustrated in Figure 5, compared with the commonly used MSE loss, our strategy is less sensitive to the samples with larger prediction error. This advantage is helpful to highlight the useful as well as accurate gradients and mitigate the negative impacts of inaccurate targets. Besides, since we divide intervals with the UEP criterion, which tries to keep the counting error cost equal for all intervals to minimize the prediction risk. We also compare the error contribution of each interval in a converged UEPNet. As shown in Figure 6c, error contributions of all intervals are approximately equal. An interesting observation is that the intervals which are closer to the middle contribute slightly more counting errors. One possible explanation is that the samples in these intervals are more difficult to learn, since they may be either overestimated or underestimated.

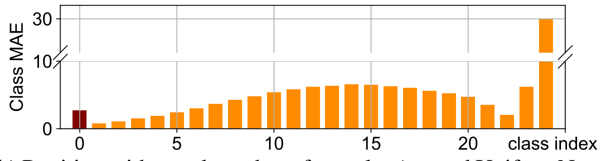
Methods	without MS		with MS	
	MAE	MSE	MAE	MSE
MSE	13.57	22.64	14.65 ↑	24.35 ↑
UniformLen	12.71	20.38	12.45 ↓	15.39 ↓
UniformNum	11.25	26.15	10.50 ↓	25.74 ↓
Ours	**7.64**	**13.18**	**6.87** ↓	**11.33** ↓

Table 7: The effect of multi-scale training.

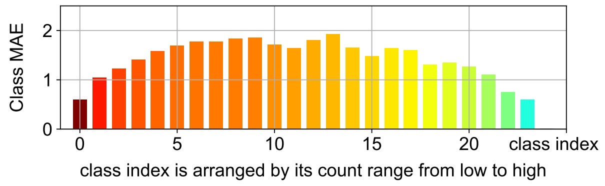
Multi-scale training is only useful for classification. Our UEPNet benefits from the data augmentation of multi-scale training. However, we find that multi-scale training is only useful for the paradigm of local count classification. As shown in Table 7, when learn to regress the exact count in each patch with MSE, the adoption of multi-scale training unexpectedly hurts the performance of the model. Although multi-scale augmentation significantly increases the diversity of samples in each interval, but also introduces much more noises due to the inaccurate targets. Consequently, using outlier-sensitive losses will force the model to overfit these noises, which leads to inferior performance. But for local count classification, the benefits of multi-scale augmentation outweigh its side effects. Additionally, compared



(a) Partition with equal length of interval (termed UniformLen).



(b) Partition with equal number of samples (termed UniformNum).



(c) Partition with the proposed UEP criterion.

Figure 6: Illustrations of counting error contributions (class MAE) from each count interval for three types of interval partition strategies. The number of samples in each count interval (class) is indicated by different colors in the top jet colormap. (a) UniformLen leads to extremely imbalanced number of samples among intervals, which makes it hard to train a well-performed classifier. (b) UniformNum leads too large interval length for certain intervals (especially for the intervals with high count), thus the misclassification of which contribute too large counting errors. (c) Our method provides a comprehensive consideration of the misclassification cost and the sample imbalance problem.

with UniformLen and UniformNum, we find that the proposed UEP criterion is helpful for maximizing the relative improvement of the multi-scale augmentation.

5. Conclusion

We propose a simple yet effective crowd counting model termed UEPNet based on the paradigm of local count classification. Specifically, we conduct deeply research on this paradigm, and propose the UEP criterion and the MCP criterion. The UEP criterion is used for the interval partition, which divides intervals with equal counting error cost for all intervals to minimize prediction risk. The MCP criterion is used for count proxy selection, which selects the average count value of samples in corresponding interval as its optimal count proxy. Additionally, we also propose the IPH to mitigate the problem of classification ambiguity for samples near the interval borders. Extensive experiments on four widely used datasets with various crowd densities demonstrate the superiority of our approach.

References

[1] Shuai Bai, Zhiqun He, Yu Qiao, Hanzhe Hu, Wei Wu, and Junjie Yan. Adaptive dilated network with self-correction supervision for counting. In *CVPR*, 2020. 1, 3, 6

[2] Zhi-Qi Cheng, Jun-Xiu Li, Qi Dai, Xiao Wu, and Alexander G Hauptmann. Learning spatial awareness to improve crowd counting. In *ICCV*, 2019. 2

[3] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018. 2

[4] Guangshuai Gao, Junyu Gao, Qingjie Liu, Qi Wang, and Yunhong Wang. Cnn-based density estimation and crowd counting: A survey. *CoRR*, 2020. 5

[5] Yutao Hu, Xiaolong Jiang, Xuhui Liu, Baochang Zhang, Jungong Han, Xianbin Cao, and David Doermann. Nas-count: Counting-by-density with neural architecture search. In *ECCV*, 2020. 2, 6

[6] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *CVPR*, 2013. 5

[7] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*, 2018. 5

[8] Xiaolong Jiang, Zehao Xiao, Baochang Zhang, Xiantong Zhen, Xianbin Cao, David Doermann, and Ling Shao. Crowd counting and density estimation by trellis encoder-decoder networks. In *CVPR*, 2019. 2

[9] Xiaoheng Jiang, Li Zhang, Mingliang Xu, Tianzhu Zhang, Pei Lv, Bing Zhou, Xin Yang, and Yanwei Pang. Attention scaling for crowd counting. In *CVPR*, 2020. 2, 6

[10] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *NIPS*, 2010. 2

[11] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *CVPR*, 2018. 2, 5, 6

[12] Liang Liu, Hao Lu, Haipeng Xiong, Ke Xian, Zhiguo Cao, and Chunhua Shen. Counting objects by blockwise classification. *TCSVT*, 2019. 1, 2, 3, 6, 7

[13] Liang Liu, Hao Lu, Hongwei Zou, Haipeng Xiong, Zhiguo Cao, and Chunhua Shen. Weighing counts: Sequential crowd counting by reinforcement learning. In *ECCV*, 2020. 6, 7

[14] Lingbo Liu, Zhilin Qiu, Guanbin Li, Shufan Liu, Wanli Ouyang, and Liang Lin. Crowd counting with deep structured scale integration network. In *ICCV*, 2019. 6

[15] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *CVPR*, 2019. 6

[16] Xiyang Liu, Jie Yang, and Wenrui Ding. Adaptive mixture regression network with local counting map for crowd counting. In *ECCV*, 2020. 2, 6

[17] Zhiheng Ma, Xing Wei, Xiaopeng Hong, and Yihong Gong. Bayesian loss for crowd count estimation with point supervision. In *ICCV*, 2019. 1, 3, 6

[18] Yunqi Miao, Zijia Lin, Guiguang Ding, and Jungong Han. Shallow feature based dense attention network for crowd counting. In *AAAI*, 2020. 1, 6

[19] Min-hwan Oh, Peder A Olsen, and Karthikeyan Natesan Ramamurthy. Crowd counting with decomposed uncertainty. In *AAAI*, 2020. 1, 6

[20] Greg Olmschenk, Hao Tang, and Zhigang Zhu. Improving dense crowd counting convolutional neural networks using inverse k-nearest neighbor maps and multiscale upsampling. *CoRR*, 2019. 1, 3

[21] Heqian Qiu, Hongliang Li, Qingbo Wu, and Hengcan Shi. Offset bin classification network for accurate object detection. In *CVPR*, 2020. 2

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[23] Jia Wan and Antoni Chan. Adaptive density map generation for crowd counting. In *ICCV*, 2019. 1, 3

[24] Haipeng Xiong, Zhiguo Cao, Hao Lu, Simon Madec, Liang Liu, and Chunhua Shen. Tasselnetv2: in-field counting of wheat spikes with context-augmented local regression networks. *Plant methods*, 2019. 2

[25] Haipeng Xiong, Hao Lu, Chengxin Liu, Liang Liu, Zhiguo Cao, and Chunhua Shen. From open set to closed set: Counting objects by spatial divide-and-conquer. In *ICCV*, 2019. 2, 6, 7

[26] Zhaoyi Yan, Yuchen Yuan, Wangmeng Zuo, Xiao Tan, Yezhen Wang, Shilei Wen, and Errui Ding. Perspective-guided convolution networks for crowd counting. In *ICCV*, 2019. 6

[27] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, 2015. 5

[28] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, 2016. 2, 4, 5