Exploiting sample correlation for crowd counting with multi-expert network

Xinyan Liu[1], Guorong Li[*1,2], Zhenjun Han[1], Weigang Zhang[3], Yifan Yang[1], Qingming Huang[1,2,4] , Nicu Sebe[5]

[1]*University of Chinese Academy of Sciences, Beijing China*
[2]*Key Lab of Big Data Mining and Knowledge Management, UCAS, Beijing, China*
[3]*Harbin Institute of Technology, Weihai, China*
[4]*Key Lab of Intelligent Information Processing, ICT, CAS, Beijing, China*
[5]*University of Trento, Trento, Italy*
liuxinyan_98@foxmail.com, {liguorong, hanzhj, qmhuang}@ucas.ac.cn, wgzhang@hit.edu.cn, yangyifan@yeah.net, sebe@disi.unitn.it

Abstract

Crowd counting is a difficult task because of the diversity of scenes. Most of the existing crowd counting methods adopt complex structures with massive backbones to enhance the generalization ability. Unfortunately, the performance of existing methods on large-scale data sets is not satisfactory. In order to handle various scenarios with less complex network, we explored how to efficiently use the multi-expert model for crowd counting tasks. We mainly focus on how to train more efficient expert networks and how to choose the most suitable expert. Specifically, we propose a task-driven similarity metric based on sample's mutual enhancement, referred as co-fine-tune similarity, which can find a more efficient subset of data for training the expert network. Similar samples are considered as a cluster which is used to obtain parameters of an expert. Besides, to make better use of the proposed method, we design a simple network called FPN with Deconvolution Counting Network, which is a more suitable base model for the multi-expert counting network. Experimental results show that multiple experts FDC (MFDC) achieves the best performance on four public data sets, including the large scale NWPU-Crowd data set. Furthermore, the MFDC trained on an extensive dense crowd data set can generalize well on the other data sets without extra training or fine-tuning.[1]

1. Introduction

Crowd counting is a task that tries to estimate the number of objects in an image, such as people, cars or animals. This task has attracted significant attention because of its various scenarios application, such as security surveillance, human traffic control, hot spot discovery etc.

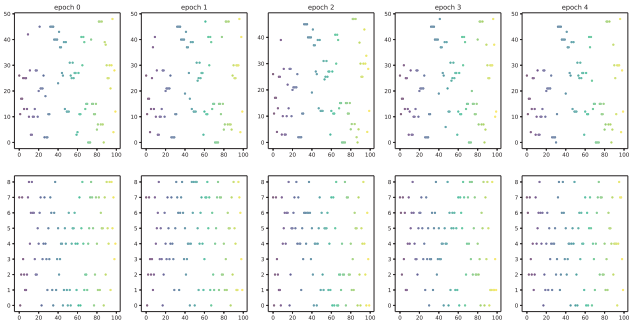

Figure 1. The stability of the category pseudo-labels during 5 epochs of alternate training. The first row is result of our method, and the second row is result of 8-way divide and grow training method from IG-CNN[16].

Earlier data sets such as UCSD data set [2], Mall data set [5], World Expo'10 data set [30] contain low density crowd image with balanced number[19]. Previous methods have achieved reliable performance on such data set [28, 29, 19, 22]. However, when it comes to more complex scenarios[32, 1, 11], the performance of existing methods will be significantly humiliated. This is because under these scenarios, there is large diversity of visual appearances because of distorted perspectives, variable scales, unbalanced distributions and wide range of brightness, *etc*. Consequently, single network based methods [12, 15, 17] may perform well in one special scenes, but worse on others.

To deal with complex scenarios, some methods [16, 12, 17, 30] explored multi-expert structure, which usually

*Corresponding author
[1]Code will be available at https://github.com/streamer-AP

includes a base feature extractors, a routing network or weighting network and multiple experts networks. Each expert was designed to handle a specific scale or density, alleviating the complexity of dealing with the diversity of scenes with only one expert. The router, essentially a classifier, selects the optimal expert for every testing sample.

To obtain effective experts, most of multi-expert counting methods [17, 16] applied a differential training technology. This technology only backwards the loss on the expert which gives the most accurate prediction of the current sample. After training, each training sample is assigned a pseudo class label, which is the index of the most suitable experts, to train the router. Existing experimental results in [16] show that the router's accuracy in assigning samples to the optimal expert network is not high. Moreover, as shown in Figure 1, the generated pseudo labels are unstable and hard to converge during differential training. This means after training, the most suitable expert for many samples is the expert that didn't use this sample to train. This goes against the original intention of the algorithm design, that is, the expert network trained with a specific set will be more suitable for the samples in this set. This phenomenon demonstrates that samples on which an expert performs the best may be not suitable for training this expert together to further boost the performance. From Figure 1, it can be seen that the category pseudo-labels obtained by our method are more stable. This means that expert networks trained using our method are indeed more suitable for the training the samples used by these networks.

Actually, since each expert is generated through fine-tuning the model on a subset of the training data, the key problem is how to divide the training set into several subsets. A good subset should bring performance improvement when fine-tune the model on it. However, it is difficult to divide the entire data set directly. We considered a easier problem, i.e. how to evaluate the similarity between two samples. We propose a novel metric to evaluate the similarity between the samples (referred co-fine-tune similarity), which can reflect the correlation between the parameters of their optimal experts. This similarity can approximately describe the model's performance improvement after fine-tuning on the subset containing these two samples. Therefore, if a cluster is composed of similar samples, it is conducive to optimize the base model in a consistent direction, generating a effective expert for all the samples in this subset. To obtain this kind of cluster, based on co-fine-tune similarity, we design a simple clustering method to find potential clusters in dense crowd data sets. Then each cluster is used to obtain the parameters of an expert. In order to select the optimal weights for testing data during the inference, we consider each cluster as a class and train a CNN classifier as router, which predicts the class label for a testing data. The prediction result is used to retrieve the optimal expert. In this way, we can dynamically select a suitable expert for the test image according to its characteristics and improve the performance greatly.

Furthermore, to reduce the storage space of the parameters and avoid over-fitting on clusters, we design a simple yet efficient crowd counting model (referred to as FDC) which has a tiny density map regressor. Using FDC as a base model, we obtain an multi-expert FDC (referred to as an MFDC) with our training strategy.

Our main contributions can be summarised as:

- We proposed a novel multi-experts training framework for crowd counting task, which exploits relations within samples. The proposed pipeline can be integrated with existing methods and improve their performance significantly.
- To obtain multiple representative weights, we develop an effective take-driven similarity and a clustering method to obtain multiple clusters of the training data. Each cluster is used to learn a set of parameters, which is effective for testing samples similar to this cluster.
- Extensive experiments are conducted on four data sets, namely, STA, STB[32], UCF-QNRF [11], and NWPU-Crowd [23], to demonstrate that the proposed method can achieve the state-of-the-art performance.

2. Related Work

Generally speaking, previous crowd counting methods can be classified into single-model methods and multi-experts methods. In this section, we analyze methods in these two trends.

2.1. Single-model methods

To deal with multiple scales, MCNN [32] implements an effective multi-branch architecture, where each branch has a distinct receptive field to adapt to targets with a specific scale range. Its success makes it a fundamental component for several works [4, 18, 14], which helps alleviate the problem of dramatic changes in scale. Compared to branches with different kernel size, Chen [4] designed a multi-column structure basing on different dilation rate. These columns share the same features from the backbone, increasing computational efficiency. MBTTBF [18] compared different feature fusion performance on counting task and design a multi-level bottom-top and top-bottom fusion method with scale aware feature extraction blocks. Context-aware [14] re-weighted features by scale from local to global guided by different pooling layers.

Some methods [9, 31, 33, 25] introduced attention mechanisms to focus on the target area in crowd counting tasks. SCAR [9] applied spatial-wise and channel-wise attention

models together to extract contextual information and background estimation. MRA-CNN [31] conducted attention on three feature maps with different resolution, and a density level estimation task guides the attention factor. ACM-CNN [33] firstly generated a rough feature map and iteratively focused and fine-tuned the density map for the image's high-density area, obtaining a more refined feature map for the high-density area. Jiang [25] designed ASNet to provide a coarse density mask and DANet to give scaling factors and several candidate density map. Comprehensive utilizing these two networks can apply specific weights for different areas of a testing sample.

These networks are helpful in multi-scale information extracting. However, the counts of objects with different density and visual appearances are hard to predict precisely by a single weight.

2.2. Multi-experts methods

There are also some works [16, 17, 12, 30] that consider training several networks independently or fine-tuning network weights for the test image during inference.

Cross Scene CNN [30] fine-tuned the network with training samples whose scenes are similar to that of the testing samples, to enhances the cross scenes performance. The similarity is calculated according to the perceptive map and density map. Nevertheless, the perceptive maps are expensive to annotate or not accessible, especially on a moving platform such as auto vehicles or drones.

MoCNN [12] applied a mixture of expert CNNs and a gating CNN to weight the importance among experts. MoCNN directly predicted the crowd counts for an image, which is more difficult than learning density map. What is more, one of the main reasons why we use the multi-expert network is that we only need to load one expert each time, but MoCNN made predictions using all the experts and then weighted sum the results, which required longer running time and computing resources.

Switch-CNN [17] designed a switch layer before the multi-column CNN, which predicted the optimal column for crops of a testing sample. Different columns are assigned with convolution kernels of various sizes to widen the margin between the results of each column, which is beneficial for training column switch layer. However, the difference between these pre-defined columns will force the switch layer focusing on scale variations of crops and ignoring the other variations. Furthermore, there are continuously changing on scales and densities, which can not be handled by a limited number of columns. Even inside a crop, multiple level scales can exist, and it is hard for a single fixed column to deal with such variety.

IG-CNN[16] applied a divide-and-grow strategy to train multi-experts network. Starting from a same base CNN density CNN generator, IG-CNN hierarchically generated

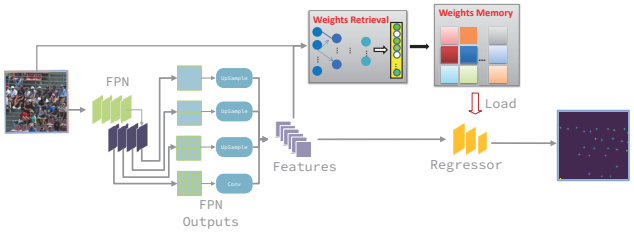

Figure 2. The structure of our proposed model. During inference, the testing sample and its feature map are used to retrieve the optimal weight from the stored weights to generate density map.

two child networks each time by fine-tuning on different samples clusters. This strategy needs reasonable metric and classification methods to split data set into clusters. IG-CNN considered samples on which the optimal loss is achieved by the same expert as a cluster during the training time. There is no guarantee that the model fine-tuned by such a cluster is the optimal one.

The main reason why the previous multi-network methods can not perform well as expected is that the previous similarity or distance metrics can not reflect the relationship between samples under crowd counting task. For multi-expert system, the key points are how to generate effective experts and assign optimal experts to testing samples. To do this, we propose a more efficient and reasonable metric to evaluate the relation between samples, and develop an effective method to train router by this metric.

3. Methods

As shown in Figure 2, our method is a multi-expert method. During training stage, a feature extractor and a density map generator are trained. Different from existing methods, we generate multiple weights for the density map generator and store them in an external memory unit. For a testing sample, the output of feature extractor concatenated with the original image's feature are used to retrieve an optimal weight from the memory unit to generate the final density map.

The overview of the training process of the proposed multi-expert method consists of 3 steps: Select informative samples and calculate the co-fine-tune similarity for training image crops; Cluster the selected samples, generate multiple experts and store the parameters of each expert; Learn the weight retrieval module for selecting the best expert for a testing sample.

3.1. Co-fine-tune similarity

Generally, given a trained crowd counting model (referred as base model) and one sample, if fine-tune this model on the given sample, the performance of the fine-tuned model on this sample as well as other similar samples will be improved. Based on the performance improvement

on the other samples, we could define a similarity between them. Specifically, let $T = \{(x_i, y_i)|i = 0, 1, 2, ..., N\}$ be the training set with N samples crops. The feature extractor and density map generator of the given model are defined as $f = \Psi(x, \theta_1)$ and $d = \Phi(f, \theta_2)$ respectively, where θ_1 and θ_2 are parameters. The loss function is denoted as $L(\hat{y}, y)$, where $\hat{y} = \Phi(\Psi(x, \theta_1), \theta_2)$. Then loss on the i-th sample (x_i, y_i) of the base model is described as $l_i = L(\Phi(f_i, \theta_2), y_i)$.

Then we fine-tune the density map regressor of the base model to get a specific and effective set of weights(denoted as θ_2^i for sample (x_i, y_i) and similar samples. The loss on the j-th sample (x_j, y_j) of the model $\Phi(f, \theta_2^j)$ could be calculated as $l_i^j = L(\Phi(f_j, \theta_2^j), y_i)$. It is expected that if sample i is similar to sample j, the fine-tuned model with the i-th sample will achieve performance improvement on j-th sample. The more similar, the larger the improvement will be. So the improvement can be seen as the similarity between them. To make the similarity symmetrical, we define the co-fine-tune similarity between i-th sample and j-th sample by:

$$s(i,j) = \begin{cases} 0, & l_i \leq l_i^j \text{ or } l_j \leq l_j^i; \\ 0.5\left(\frac{l_i}{l_i^j} + \frac{l_j}{l_j^i}\right) - 1, & \text{otherwise.} \end{cases} \quad (1)$$

For two samples, the proposed co-fine-tune similarity reflects mutual improvement between two samples. If the performance on either of the two is improved after fine-tuning the base model on the other one, the similarity between them will be positive, and the greater the ratio of mutual improvement, the greater the similarity is.

However, it is quite time-consuming to calculate the co-fine-tune similarity of all the samples. Intuitively, hard samples are important, but we discovered that there were some unstable samples, on which the loss of the base model was not stable during the process of training. Although some of them are not hard samples, fine-tuning on these unstable samples in the data set will significantly decreased the loss on these samples. So, we calculate co-fine-tune similarity among unstable samples rather than hard samples. Besides, as the loss on stable samples varies little during training, fine-tune on them will make little difference to the parameters. Therefore, the co-fine-tune similarity between these samples can be directly considered as 0.

To evaluate the instability of samples during the training process, we adopt the sequence and reversal test by [6] and only considering the downward trend of the loss function. We define an indicator $I(i, t)$ in Eq (2) for downward trend on i-th sample in m-th epoch, described as:

$$I(i,m) = \begin{cases} 0, & L(\widehat{y}_i^m, y_i) > L(\widehat{y}_i^{m-1}, y_i) + \epsilon; \\ 1, & \text{otherwise.} \end{cases} \quad (2)$$

where hyper-parameter ϵ is used to regulate the tolerance to slight changes. The stability of sample x_a during M epochs can be calculated by Eq (3).

$$\widehat{J}_i = \frac{\sum_{m=1}^{M} I(i,m)}{M} \quad (3)$$

The instability of the i-th sample is larger if $\widehat{J}_i$ is closer to 1. The samples whose instability is larger than a threshold η are chosen to form an unstable set denoted as $\mathcal{Q}$. We only calculate the co-fine-tune similarity of all the samples in $\mathcal{Q}$. For simplicity, in the following part, $s(i, j)$ means the similarity between the i-th sample and j-th sample in $\mathcal{Q}$.

3.2. Clustering methods

To reduce the time cost of fine-tuning in real-time, we designed a clustering method to classify unstable samples into several clusters, and each cluster is used to obtain a set of weights. We design of the clustering algorithm according to two principles. First, the co-fine-tune similarity of all samples within each cluster should be positive. Second, the number of clusters should be as small as possible to reduce the space needed for weights memory. Motivated by DBSCAN [7], we design a heuristic clustering algorithm, whose detail is described in Algorithm 1.

The clustering results are denoted as $S_1, S_2, \cdots, S_K$, where S_i denotes a cluster and K denotes the number of obtained clusters by the proposed algorithm.

For all the samples that are not in $\mathcal{Q}$, we consider them as one cluster denoted as S_0. Then we use every cluster to fine-tune the density map regressor of the base model and obtain $K + 1$ sets of weights denoted as $\hat{\theta}_2^k, k = 1, \cdots, K$.

3.3. Weight retrieval methods

To retrieve the optimal weight for a testing sample, we consider each cluster in $S_1, \cdots, S_K$ as one class and train a multi-class classifier. However, a sample belonging to one cluster may have positive co-fine-tune similarity with part of samples in other clusters. So instead of simply using the hard label, we introduce a soft label which is calculated by the average similarity between samples in clusters.

We adopt ResNet-18 [10] as the backbone of our classifier. The inputs consist of two parts: the original image and the corresponding output of features extractors from the base model. The original image is aligned to the same size of features extractor's output by a shallow CNN-Pool-CNN structure. The cross-entropy loss is adopted for training our multi-class classifier. For a testing sample x, the prediction result of our classifier is represented as $\mu = (\mu_1, \cdots, \mu_K)$, where μ_i is the probability of x belonging to cluster S_i. If every element of μ is small, then the probability of data belonging to each class is small and we consider it comes from cluster S_0. We define a threshold, and the final prediction result is given by Eq (4), where N is the number of clusters.

Algorithm 1: The Proposed Clustering Method

Input: similarity matrix $s(i,j)$,$\mathcal{Q}$
Output: K clusters $\{S_i \| i = 1, \cdots, K\}$
Initialize $\mathbf{V}$ as a zero vector;
Copy unstable samples in $\mathcal{Q}$ to U;
$k \leftarrow 0$;
for u *in* U **do**
$T(u) \leftarrow \sum_{v \in \mathcal{Q}} s(u.idx, v.idx)$; /* Calculate sum of its similarity to all other samples in $\mathcal{Q}$ */
end for
Sort samples in U according to T descendingly;
for u *in* U **do**
if $\mathbf{V}(u) == 1$ **then**
continue;
end if
$\mathbf{V}(u) \leftarrow 1$;
$k \leftarrow k + 1$;
$S_k \leftarrow \{u\}$;
Copy unstable samples in $\mathcal{Q}$ to V;
Sort V according to $s(u.idx, v.idx)$ descendingly;
for v in V **do**
if $s(u.idx, v.idx) \leq 0$ **or** $\mathbf{V}(v) == 1$ **then**
continue;
else
if v is similar to all samples in S_k **then**
$S_k \leftarrow S_k \cup \{v\}$;
$\mathbf{V}(v) \leftarrow 1$;
end if
end if
end for
end for

$$d(x) = \begin{cases} 0, & \max\{\mu\} \leq \frac{1}{N}; \\ \arg\max_j \{\mu\}, & \text{otherwise.} \end{cases} \tag{4}$$

Then for each crop of the testing sample x, the weights obtained by $S_{d(x)}$ will be loaded to the density map generator for prediction a refined density map.

3.4. The proposed Base Model

The density map generator will be fine-tuned several times both in the process of co-fine-tuning calculation and clustering. In addition to the need for large parameter storage space, a model with a large density map generator is easy to over-fit if the training set is small. To alleviate these problems, we design a simple base model called FDC, which consists of the standard Feature Pyramid Networks (FPN) [13] as a basic feature extractor and dilated convolution as the density map generator.

The FPN in FDC can flexibly adopt different networks as backbone, such as ResNet-18, ResNet-34, ResNet-50, *et al*. In our method, we mainly use ResNet-18, which is efficient enough. To align the output of FPN, we added de-convolution layers as up-samplers. Compared to state-of-the-art methods, the density map regressor of FDC has fewer parameters but is still efficient enough.

During the training process, patches are cropped from the original image with sizes of 224×224, 448×448, 896×896, and resized into 224×224. Four feature maps are generated by FPN with size from 7×7 to 56×56. Then these feature maps are up-sampled by multi de-convolution layers with the stride of 2 to generate feature maps with the same size of 56×56. Then these feature maps are concatenated and fused by two convolution layers with dilation of 3 to generate the output density map.

4. Experiments

We evaluate our algorithm against previous state-of-the-art methods on four public open-view dense crowd data sets, namely, STA, STB, [32], UCF-QNRF [11], NWPU-Crowd [23]. To demonstrate the effectiveness of the proposed multi-expert strategy, we also apply the proposed framework on CSRNet [4] and MCNN [32], and obtain a multi-expert version of MCNN (referred as M-MCNN) and a multi-expert version CSRNet (referred as M-CSRNet). Our proposed FDC with ResNet-18 is denoted as FDC-18 for short, and the multi-expert FDC-18 is referred to as MFDC-18. Along with these experiments, we also tested the cross data set performance.

The implementation of our model is conducted by four GTX 2080Ti GPUs. For training of FDC-18, we set the batch size to 128, inner iterations to 500, and apply Adam as the optimizer with a fixed learning rate 10^{-5}. For fine-tuning on each cluster, we set the inner batch size to 16, the max inner iterations to 50, and the learning rate to 10^{-6} decaying by the mean absolute error. If the cluster's number of samples is smaller than 100, augmentation will be conducted to complete the number of samples to 100. Random horizontal flip, random blur, random contrast and brightness are augmentations applied in our experiments.

4.1. Evaluation metrics

Mean Absolute Error (MAE) and Mean Square Error (MSE) are two main metrics for evaluating the performance of crowd counting algorithms, and they are defined as:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |C_i - C_i^{gt}| \tag{5}$$

$$\text{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} |C_i - C_i^{gt}|^2} \tag{6}$$

where N is the number of images in test data set, C_i represents the predicted count, which can be calculated by in-

Methods	MAE(A)	MSE(A)	MAE(B)	MAE(B)
MCNN [32]	110.6	171.1	26.4	41.3
IG-CNN[16]	72.5	118.2	13.6	21.1
CSRNet[4]	68.2	115.0	10.6	16.0
SFCN-101[24]	64.8	107.5	7.6	13.0
CAN[14]	62.3	100.0	7.8	12.2
DM-Count[22]	59.7	95.7	7.4	11.8
SDCNet[27]	58.3	95.0	**6.7**	10.7
M-MCNN	94.1	127.5	24.1	36.4
M-CSRNet	60.1	98.7	7.2	11.5
FDC-18	65.4	109.2	11.4	19.1
MFDC-18	**55.4**	**91.3**	6.9	**10.3**

Table 1. Comparison with other state-of-the-art crowd counting methods on STA and STB.

Methods	MAE	MSE
MCNN[32]	277	426
Switch-CNN[17]	228	445
CAN[14]	107	183
CSRNet[4]	98.2	157.2
SDCNet[27]	97.7	167.6
DM-Count[22]	85.6	148.3
SS-DCNet(cls)[26]	81.9	143.8
M-MCNN	234.1	381.8
M-CSRNet	83.1	144.6
FDC-18	93.0	157.3
MFDC-18	**76.2**	**121.5**

Table 2. Performance of our proposed methods compared to previous state-of-the-art methods on UCF-QNRF.

tegrating over the pixels of the predicted density maps, and C_i^{gt} is the ground truth.

On some data sets with fine-grained scene labels, we also test scenes-wise metrics, which are defined by:

$$\text{Avg.MAE} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{N_i}\sum_{j=1}^{N}|C_{i,j} - C_{i,j}^{gt}| \tag{7}$$

where M is the number of scenes, and N_j means the number of samples in j-th scene.

4.2. Performance on dense crowd data set

ShanghaiTech dataset [32] includes two parts, STA and STB. We used official division of the data set To expand the data set and applied augmentation technologies provided in [3] (including horizontal flip, blur, random cropping and resize, optical distortion and random contrast and brightness) and generated 6000 crops using the training data sets. To make the best performance, for STA, the threshold η is set to 0.36 and 1000 unstable crops are selected as set $\mathcal{Q}$ and split into 97 clusters. For STB, η is 0.42 and 1500 unstable crops selected. The performance is shown in Table 1. With the help of our multi-expert strategy, both M-MCNN and M-CSRNet achieve considerable improvement. Although the density map regressor is simple, the performance of FDC-18 is comparable to that of some larger models (e.g. CAN [14]). Thanks to multi-expert, MFDC-18 improves the MAE of FDC-18 by 15.3% on STA, and 39.4% on STB. Both of them get the lowest MAE as far as we know.

UCF-QNRF [11] is a large counting data set. By the same augmentation methods mentioned in experiments on STA and STB, we cropped 24020 samples from training images. The threshold η is set to 0.41 and 4000 unstable crops are selected as set $\mathcal{Q}$ and split into 357 clusters. The performance is shown in Table 2, and MFDC-18 obtained the best performance. Compared with the second-best method (SS-DCNet(cls) [26]), MFDC-18 improves MAE by 7%. Besides, Compared with the corresponding based models,

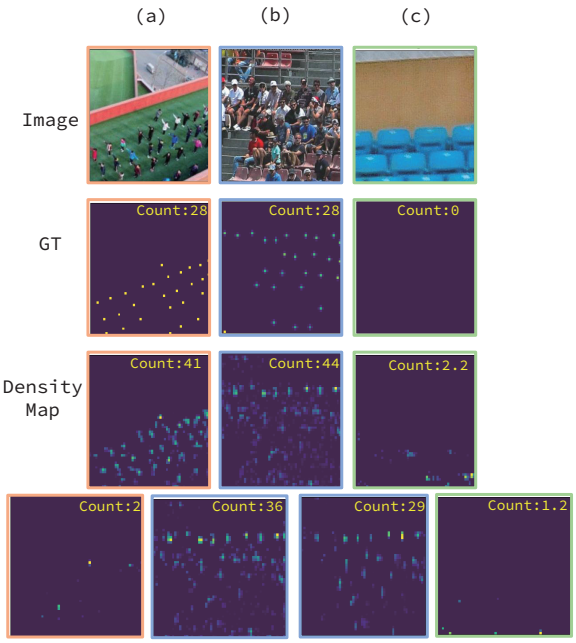

Figure 3. From the top to the third row, they are original images, ground truth,and density maps predicted by FDC-18. Image (a) and (b) have similar density map, while image (b) and (c) are similar according to their co-fine-tune similarity. In the bottom row, the first image is the predicted density map of image (a) with the fine-tuned model on image (b). The second and the third one are the prediction results of image (b) with the fine-tuned model on image (a) and (c) respectively. The last one is the prediction results of image (c) with the fine-tuned model on image (b).

M-MCNN and M-CSRNet decrease MAE by 15.5% and 15.3% respectively, while MFDC-18 decreases MAE and MSE by 18.1% and 22.8% respectively. The improvement of MFDC-18 is the most significant. We think the reason is that the regressor of MCNN [32] is too simple to fit well on each cluster, while the regressor of CSRNet is too complex, leading to overfitting, and the size of FDC's regressor

Methods	O_MAE	O_MSE	O_NAE	Avg.MAE[S]	Avg.MAE[L]
MCNN[32]	232.5	714.6	1.063	1171.9	220.9
CSRNet [4]	121.3	387.8	0.604	522.7	112.0
SCAR[8]	110.0	495.3	0.288	718.3	102.3
CAN[14]	106.3	386.5	0.295	612.2	102.1
SFCN101[24]	105.7	424.1	0.254	712.7	106.8
S-DCNet[27]	90.2	370.5	0.285	567.8	82.9
DM-Count[22]	88.4	388.6	**0.169**	498.0	88.0
FDC-18	119.39	380.6	0.34	642.7	105
MFDC-18	**74.7**	**267.9**	0.184	**412.2**	**67.6**

Table 3. Couting performance of the proposed methods and the previous state-of-the-art methods on NWPU-Crowd test set.

Division criteria	Luminance level			Density level				
Methods	[0,0.25]	[0.25,0.5]	[0.5,0.75]	0	(0,100]	(100,500]	(500,5000]	(5000,)
DM-Count[22]	203.64	88.07	61.195	146	**7.63**	**31.19**	228.70	2075.78
FDC-18	206.13	119.78	97.1	81.37	20.4	59.57	295.53	2756.7
MFDC-18	**138.52**	**75.24**	**59.57**	**5.68**	8.55	33.3	**215.97**	**1797.7**

Table 4. Comparisons on NWPU-Crowd official subsets with different luminance and density levels.

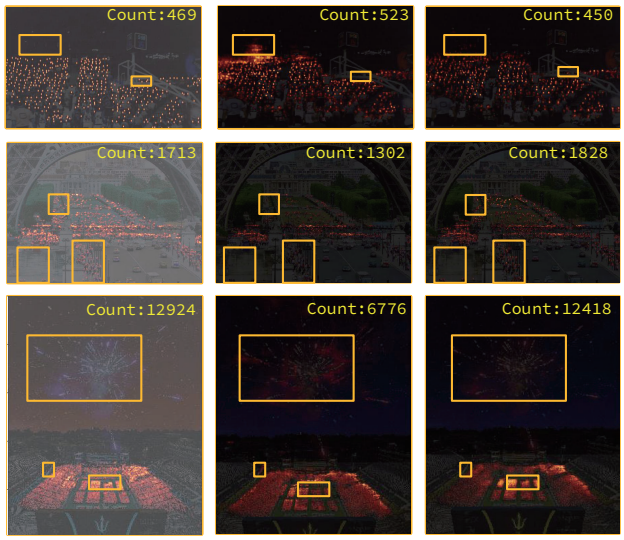

Figure 4. Performance of the FDC and the MFDC methods on NWPU-Crowd data set. From left to right, they are ground truth, density map predicted by FDC, density map predicted by MFDC

is suitable for multi-expert strategy.

NWPU-Crowd [23] is a large-scale crowd counting data set with 5109 high resolution images. In addition to the head markers, the bounding boxes, the brightness labels and density labels of images are also annotated. In experiments, the threshold η is set to 0.64 and 10000 unstable crops are selected as set $\mathcal{Q}$, split into 592 clusters, and the performance on NWPU-Crowd test set is shown in Table 3. MFDC-18 largely outperforms all compared approaches in four evaluation metrics. To better illustrate the proposed co-fine-tune similarity, Figure 3 shows some qualitative results. The proposed co-fine-tune similarity is more effective for describing the relation between samples. In Figure 4, we show performance of the FDC and the MFDC methods on three typical images from NWPU-Crowd data set with different scenes, brightness, and crowd distribution. MFDC significantly improved the prediction results of FDC .

Performances of FDC-18 and MFDC-18 are compared to previous state-of-the-art DM-Count in different crowd level and luminance level in Table 4. The results show that MFDC-18 improves the MAE at all luminance levels, especially when it comes to images with a low luminance level. Furthermore, we observe that MFDC-18 is powerful not only to identify the sample's without crowd but also predict accurate counts with high-density samples. The multi-expert model significantly improves the adaptability of the model in various situation.

4.3. Ablation Study

In this section, we conduct several experiments to study the effect of different sample selection methods, the transferability of the proposed method and training time.

Sample Selection Method. There are two methods for selecting subset $\mathcal{Q}$. The first method is to select hard samples on which the loss of the base model is high. The other one is to select samples with higher instabilities. We compared these two methods using the FDC trained by the NWPU-Count data set. The results are shown in Figure 5. When selecting a small number of samples, the performance of the two methods is similar. However, when selecting more samples, the performance of the second method is better. The reason is that when selecting too many samples, their hardness is not as high as expected. While selecting by instability can get more effective samples.

Methods	MFDC target		MFDC source		FDC target		FDC source		FDC fine-tune	
data set	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ST Part_A	55.4	91.3	56.9	94.7	65.4	109.2	93.5	153.1	63.1	94.7
UCF-QNRF	76.2	121.5	81.0	145.9	93.0	157.3	255.9	432.2	90.4	133.2
JHU-Crowd	58.1	221.9	72.3	241.2	77.8	263.1	167.6	391.2	78.6	271.5

Table 5. Analysis of the transferability of the proposed method. Target means using the target data to train the model, while source means using weights trained on NWPU-Crowd data set without any extra training. Fine-tune means using weights pre-trained on NWPU-Crowd data set, then fine-tuned on target data set.

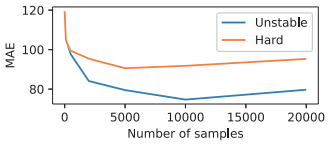

Figure 5. Comparisons between the two sample selection methods.

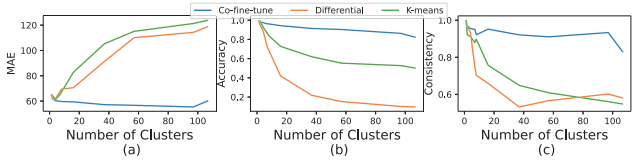

Figure 6. The orinate of (a), (b), (c) are MAE, Classification Accuracy, Label Consistency and their abssisa is the number of clusters.

Effect of the number of clusters. To verify whether the obtained cluster number if the optimal, we change it by merging the smallest clusters or splitting the biggest clusters by KNN on STA. As shown in Figure 6(a), when reduce or increase the number of the clusters, the performance degrades. This proves that the number of clusters adaptively decided by our clustering method is the best.

Consistency and Classification Accuracy . For a training sample, if the best expert for it is the expert obtained by fine-tuning the density map regressor of the base model on the cluster it belonging to, we think this sample is label consistent. We introduce label consistency as the rate of the training samples that are label consistent. The classification accuracy means the accuracy of the router selecting the best experts. We conduct ablation study to analyze label consistency and classification accuracy of three different strategies. The results are shown in Figure 6(b) and Figure 6(c). We can see that the subsets generated by proposed methods are more consistency and easier for classification, resulting in lower MAE as number of experts increase.

Transferability. We also conduct transfer experiments from NWPU-Crowd data set to STA, UCF-QNRF, JHU-Crowd++[20, 21]. The MFDC-18 and FDC-18 were trained on NWPU-Crowd data and then tested on the other data sets (referred to as target data set) without any extra training or fine-tuning. The result is shown in Table 5. It can be seen that MFDC-18 trained on NWPU can achieve better performance on target data set than FDC-18 trained on with all possible strategies, i.e. trained on the NWPU, trained on the target data set, trained on the source data set and fine-tune on the target. Compared with MFDC trained on the target data set, the performance of MFDC trained on NWPU-Crowd dropped only a little, which demonstrates that the proposed multi-expert method can endow a network more powerful generalization ability.

Training time Complexity is essential when training multi-expert networks. With constant θ training epochs, the training time of single model is $\mathcal{O}(n)$ for n samples, and for MFDC, it is $\mathcal{O}(n^2)$. More precisely, the total training time of FDC is $\alpha n\theta$, where α represents the average training time of one sample. For MFDC, the training time is formulated as $f(n) = \alpha n\theta + 10\alpha\rho n + \alpha\rho^2 n^2 + f_c(n)$, where ρ denotes the proportion of unstable samples in all samples. In our experiments, ρ ranges from 0.15 to 0.2, and the training time of weight classifier $f_c(n)$ is shorter than $\alpha n\theta$.

5. Conclusion

This paper introduces a novel multi-experts training methods for crowd counting tasks basing on the co-fine-tune similarity, which estimate the similarity between the optimal experts for samples. Based on it, representative clusters of training data are generated to obtain several sets of experts for density map generation. During inference, testing samples with different characteristics are treated differently by selecting the specific expert fine-tuned with similar training samples. Also, this method can work together with several previous state-of-the-art single methods.

In order to better demonstrate the effectiveness of the proposed multi-expert strategy, we propose a simple FDC network. Experiments on several crowd data sets show that the proposed multi-expert method significantly improves the performance of base models, especially FDC, and can achieve state-of-the-art performance on all of those data set.

Acknowledgements This work was supported in part by the Italy–China Collaboration Project TALENT under Grant 2018YFE0118400; in part by the National Natural Science Foundation of China under Grant 61620106009, Grant 61772494, Grant 61931008, Grant 61836002, and Grant 61976069; in part by the Youth Innovation Promotion Association CAS; and in part by the Fundamental Research Funds for Central Universities.

References

[1] CrowdBenchmark All. Crowd benchmark. 1

[2] Chan Aotoni, B, Liang Zhang-Sheng John, and Vasconcelos Nuno. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, 2008. 1

[3] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2):125, Feb 2020. 6

[4] Xinya Chen, Yanrui Bin, Nong Sang, and Changxin Gao. Scale pyramid network for crowd counting. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, pages 1941–1950. Institute of Electrical and Electronics Engineers Inc., mar 2019. 2, 5, 6, 7

[5] Zhongwei Cheng, Lei Qin, Qingming Huang, Shuicheng Yan, and Qi Tian. Recognizing human group action by layered model with multiple cues. *Neurocomputing*, 136:124–135, 2014. 1

[6] Alfred Cowles and Herbert E. Jones. Some A Posteriori Probabilities in Stock Market Action. *Econometrica*, 1937. 4

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 1996. 4

[8] Junyu Gao, Wei Lin, Bin Zhao, Dong Wang, Chenyu Gao, and Jun Wen. C^3 framework: An open-source pytorch code for crowd counting. *arXiv preprint arXiv:1907.02724*, 2019. 7

[9] Junyu Gao, Qi Wang, and Yuan Yuan. Scar: Spatial-/channel-wise attention regression networks for crowd counting. *Neurocomputing*, 363:1–8, 2019. 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 4

[11] 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, 2018. 1, 2, 5, 6

[12] Shaohei Kumagai, Kazuhiro Hotta, and Takio Kurita. Mixture of counting cnns: Adaptive integration of cnns specialized to specific appearance for crowd counting. 2017. 1, 3

[13] Tsung-Yi Lin, Dollár Piotr, Girshick Ross, Kaiming He, Hariharan Bharath, and Belongie Serge. Feature pyramid networks for object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 5

[14] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, 2019. 2, 6, 7

[15] Daniel Oñoro-Rubio and Roberto J. López-Sastre. Towards perspective-free object counting with deep learning. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 615–629, Cham, 2016. Springer International Publishing. 1

[16] Deepak Sam, Neeraj Sajjan, R. Babu, and Mukundhan Srinivasan. Divide and grow: Capturing huge diversity in crowd images with incrementally growing cnn. pages 3618–3626, 06 2018. 1, 2, 3, 6

[17] Deepak Babu Sam, Shiv Surya, and R. Venkatesh Babu. Switching convolutional neural network for crowd counting. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 2017-January, 2017. 1, 2, 3, 6

[18] Vishwanath Sindagi and Vishal Patel. Multi-level bottom-top and top-bottom feature fusion for crowd counting. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 2

[19] Vishwanath A. Sindagi and Vishal M. Patel. A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107:3–16, 2018. Video Surveillance-oriented Biometrics. 1

[20] Vishwanath A Sindagi, Rajeev Yasarla, and Vishal M Patel. Pushing the frontiers of unconstrained crowd counting: New dataset and benchmark method. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1221–1231, 2019. 8

[21] Vishwanath A Sindagi, Rajeev Yasarla, and Vishal M Patel. Jhu-crowd++: Large-scale crowd counting dataset and a benchmark method. *Technical Report*, 2020. 8

[22] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai. Distribution matching for crowd counting. In *Proceedings of Advances in Neural Information Processing Systems*, 2020. 1, 6, 7

[23] Qi Wang, Junyu Gao, Wei Lin, and Xuelong Li. Nwpu-crowd: A large-scale benchmark for crowd counting and localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2, 5, 7

[24] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8198–8207, 2019. 6, 7

[25] Jiang Xiaoheng, Zhang Li, Xu Mingliang, Zhang Tianzhu, Lv Pei, Zhou Bing, Yang Xin, and Pang Yanwei. Attention scaling for crowd counting. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4705–4714, 2020. 2, 3

[26] Haipeng Xiong, Hao Lu, Chengxin Liu, Liu Liang, Zhiguo Cao, and Chunhua Shen. From open set to closed set: Counting objects by spatial divide-and-conquer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8362–8371, 2019. 6

[27] 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, 2019. 6, 7

[28] Yifan Yang, Guorong Li, Dawei Du, Qingming Huang, and Nicu Sebe. Embedding perspective analysis into multi-column convolutional neural network for crowd counting.

IEEE Transactions on Image Processing, 30:1395–1407, 2021. 1

[29] Yifan Yang, Guorong Li, Zhe Wu, Li Su, Qingming Huang, and Nicu Sebe. Reverse perspective network for perspective-aware object counting. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4373–4382, 2020. 1

[30] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June-2015, 2015. 1, 3

[31] Youmei Zhang, Chunluan Zhou, Faliang Chang, and Alex C. Kot. Multi-resolution attention convolutional neural network for crowd counting. *Neurocomputing*, 329, 2019. 2, 3

[32] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 589–597. IEEE Computer Society, dec 2016. 1, 2, 5, 6, 7

[33] Zhikang Zou, Yu Cheng, Xiaoye Qu, Shouling Ji, Xiaoxiao Guo, and Pan Zhou. Attend to count: Crowd counting with adaptive capacity multi-scale cnns. *Neurocomputing*, 367:75 – 83, 2019. 2, 3