Weighing Counts: Sequential Crowd Counting by Reinforcement Learning

Liang Liu[1⋆], Hao Lu[2], Hongwei Zou[1], Haipeng Xiong[1], Zhiguo Cao[1⋆⋆], and Chunhua Shen[2]

[1] School of Aritifical Intelligence & Automation, Huazhong University of Science & Technology, China
[2] The University of Adelaide, Australia
{wings, zgcao}@hust.edu.cn

Abstract. We formulate counting as a sequential decision problem and present a novel crowd counting model solvable by deep reinforcement learning. In contrast to existing counting models that directly output count values, we divide one-step estimation into a sequence of much easier and more tractable sub-decision problems. Such sequential decision nature corresponds exactly to a physical process in reality—scale weighing. Inspired by scale weighing, we propose a novel 'counting scale' termed LibraNet where the count value is analogized by weight. By virtually placing a crowd image on one side of a scale, LibraNet (agent) sequentially learns to place appropriate weights on the other side to match the crowd count. At each step, LibraNet chooses one weight (action) from the weight box (the pre-defined action pool) according to the current crowd image features and weights placed on the scale pan (state). LibraNet is required to learn to balance the scale according to the feedback of the needle (Q values). We show that LibraNet exactly implements scale weighing by visualizing the decision process how LibraNet chooses actions. Extensive experiments demonstrate the effectiveness of our design choices and report state-of-the-art results on a few crowd counting benchmarks, including ShanghaiTech, UCF_CC_50 and UCF-QNRF. We also demonstrate good cross-dataset generalization of LibraNet. Code and models are made available at https://git.io/libranet

Keywords: Crowd Counting · Reinforcement Learning

1 Introduction

Counting is sequential decision process by nature. Dense object counts are not inferred by humans with a simple glance [4]. Instead humans count objects in a sequential manner, with initial fast counting on apparent objects (large sizes and clear appearance) and gradually slow counting on objects that are hard to recognize (small sizes or blurred appearance). Such a sequential decision behavior

⋆ L. Liu and H. Lu contributed equally
⋆⋆ Corresponding author

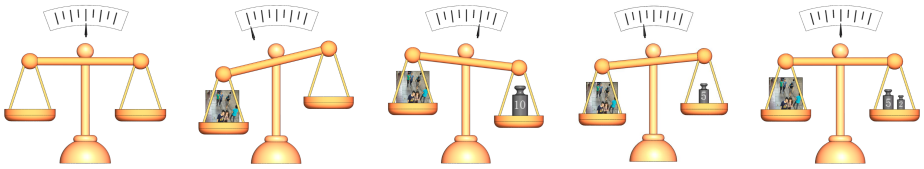

Fig. 1: Counting scale. We implement crowd counting as scale weighing. By virtually placing a crowd image (with 7 people) on the scale, if placing a 10 g weight on the scale pan, the scale will lean to the right; if exchanging the 10 g weight to 5 g, the scale instead will lean to the left. Finally by adding another 2 g weight, the scale is balanced. The total weights on the scale can therefore indicate the number of crowd.

can be modeled by a physical process in reality—scale weighing. In scale weighing, it is easy to choose a weight when the weights placed on the scale are far from the true weight of the stuff. When placed weights are close to the true weight, small and light weights are carefully chosen until the needle indicates the balance. This process decomposes a difficult problem into a series of much more tractable sub-problems.

Following the same spirit of human counting and scale weighing, we formulate counting as a sequential decision problem and implement it as scale weighing. Indeed counting objects is like weighing stuff. In the context of crowd counting shown in Fig. 1, the 'stuff' is a crowd image, and the 'weights' are a series of pre-defined value operators. We repeatedly choose counting 'weights' to approximate the ground-truth counts until the scale is balanced. The final image count is simply a summation of placed 'weights'.

The sequential decision nature of scale weighing makes it suitable to be described by a reinforcement learning (RL) task. We hence propose a Deep Q-Network (DQN) [29]-based solution, LibraNet[3], to implement scale weighing and apply it to crowd counting as a 'counting scale'. In particular, given a 'stuff', LibraNet outputs a combination of weights step by step. In each step, a weight (action) is chosen from the weight box (the pre-defined action pool) or removed from the scale pan according to the feedback of the needle (Q values that indicate how to choose the next action). The weighing process continues until LibraNet chooses the 'end' operator. The 'stuff' is the image feature encoded from a crowd image, and the 'end' condition meets when the summation of the weights equals/approximates to the ground-truth people count.

We visualize how LibraNet works and illustrate that LibraNet exactly implements scale weighing. We show through extensive experiments why our choices in designing reward function work well, that LibraNet can be used as a plug-in to existing local counts models [47, 19], and that LibraNet achieves state-of-the-art performance on three crowd counting datasets, including ShanghaiTech [52], UCF_CC_50 [11], and UCF-QNRF [12]. We also report cross-dataset performance to verify the generalization of LibraNet.

[3] The naming of LibraNet is inspired by the zodiac sign.

In summary, we show that counting can be interpreted as scale weighing and we implement scale weighing with LibraNet. To our knowledge, LibraNet is the first approach that uses RL techniques to solve crowd counting.

1.1 Related Work

Crowd Counting. Crowd counting is often tackled as a dense prediction task [24, 25]. Solutions range from early attempts that detect pedestrians [7], regress image counts [3], estimate density maps [16], predict localized counts [5], to recent deep learning-based density maps estimation [17], redundant counts regression [6, 23], instance blobs localization [15] and count intervals classification [19, 48].

Since detection typically failed on small and dense people, regression-based approaches [3, 32] were proposed. While early methods alleviated the issues of occlusion and clutter, they ignored spatial information because only the global image count was regressed. This situation eased when the concept of density map was introduced in [16]. Chen *et al.* [5] also introduced localized count regression by mining local feature importance and sharing visual information among spatial regions.

With the success of Deep Convolutional Neural Networks (DCNNs), deep crowd counting models emerged. [45] applied a CNN to crowd counting by global count regression. [51] presented a switchable training scheme to estimate the density map and the global count. By contrast, works of [6, 23] adopted redundant counting where local patches were densely sampled in a sliding-window manner during training, and the image count was obtained by normalizing redundant local counts at inference time. Authors of [20] employed a CRF-based structured feature enhancement module and a dilated multiscale structural similarity loss to address scale variations of crowd. To alleviate perspective distortion, the work in [35] integrated perspective information into density regression and proposed a PACNN for efficient crowd counting. In [15] a network is trained to output a single blob for each person for localization. The work in [44] optimized a residual signal to refine the density map. Instead of direct regression, authors of [19, 48] reformulated it as a classification problem by discretizing local counts and classifying count intervals.

Most existing models generate crowd counts in one step. This renders difficulties in correcting under- or over-estimated counts. Despite that there exists a method that recurrently refines density map with a spatial transformer network [21], it does not decompose a hard task into a sequence of easy sub-tasks and does not fully leverage the advantage of sequential counting.

Deep Reinforcement Learning. RL [8, 31] is one of the fundamental machine learning paradigms. It includes several elements, namely, agent, environment, policy, state, action, and reward. It aims to learn policies such that an agent can receive the maximum reward when interacting with the environment. Since the work of [28] introduced deep learning into RL, it has received extensive studies [27, 29, 34, 46]. In particular, RL achieved breakthroughs in a few areas such as go [37] and real-time strategy games [30, 43]. Recently some deep RL-based methods

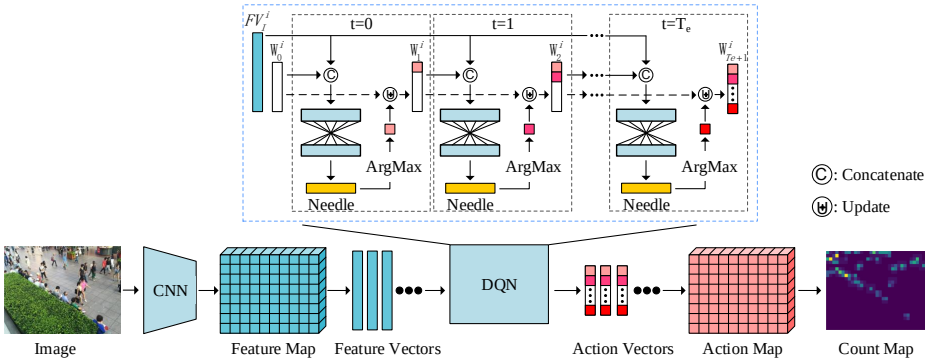

Fig. 2: Overview of LibraNet. A CNN backbone first extracts the feature map FV_I of an input image I, then each element FV_I^i of FV_I is sent to a DQN. In DQN, FV_I^i and a weighing vector W_t^i are concatenated and sent to a 2-layer MLP. The output of MLP is a 9-dimensional Q value vector. We choose an action with the maximum Q value, and update W_{t+1}^i per Eq. (4). This process repeats until the model chooses 'end' or exceeds the predefined maximum step. The output action vectors can be converted to count intervals by Eq. (5), and the intervals are further remapped to a count map with inverse discretization [19]. The image count of I is acquired by summing the count map.

were also proposed to tackle computer vision tasks, such as object localization [2] and instance segmentation [1]. However, these RL practices in computer vision cannot be directly transferred to crowd counting. A main reason is that there is no principled way to reformulate counting into a sequential decision problem suitable for RL. Inspired by scale weighing, we fill this gap and present the first deep RL-based approach to crowd counting.

2 Sequential Crowd Counting by Reinforcement Learning

Here we explain LibraNet in detail. Sec. 2.1 introduces the formulation of sequential counting. Sec. 2.2 shows how to deal with this sequential task with Q-learning. Sec. 2.3 explains the network architecture, and Sec. 2.4 presents implementation details. An overview of our method is shown in Fig. 2.

2.1 Generalized Local Count Modeling

Despite that most deep counting networks treat density maps as the regression target [12, 26, 33, 45, 52], there is another line of works pursuing the idea of local count modeling and also reporting promising results [6, 19, 23, 48]. LibraNet follows this local count paradigm but operates in a sequential manner. In what follows, we present a generalized perspective of local count modeling and show how we reformulate them into sequential learning.

Local Count Regression. Some previous works [6, 23, 47] consider counting a problem of local count regression, which densely samples an image into a series of local patches then estimates the per-patch count directly. It amounts to the following optimization problem

$$\min_{\theta} \sum_{i \in I} \left| G\left(i\right) - N_R^{\theta}\left(i\right) \right|, \tag{1}$$

where I is the input image and i denotes the local patch sampled from I, $G\left(i\right)$ returns the ground truth count given i, and N_R^{θ} is a regression network parameterized by θ.

Local Count Classification. Inspired by local count regression, counting is further formulated as a classification problem [19, 48] where local patch counts are discretized into count intervals. This process is defined by

$$\min_{\theta} \sum_{i \in I} \left| G\left(i\right) - \mathrm{ID}\left(\arg\max_{c} N_C^{\theta}\left(i, c\right) \right) \right|, \tag{2}$$

where N_C^{θ} is a classification network parameterized by θ, c is the number of count intervals, and $\mathrm{ID}(\cdot)$ defines an inverse-discretization procedure that recovers the count value from the count interval [19]. More details about discretization and inverse-discretization can be referred to Supplementary Materials.

Local Counting by Sequential Decision. Motivated by scale weighing, counting can be transformed into a sequential decision task. We call this a *weighing task*. Instead of estimating a count value or a count interval directly, the weighing task sequentially chooses a value operation in each step from a pre-defined action pool. The sequential process terminates when the agent chooses the 'ending' operation or exceeds the maximum step allowed. This task is defined by

$$\min_{\theta} \sum_{i \in I} \left| G\left(i\right) - \sum_{t=0}^{T_e} \arg\max_{a} N_E^{\theta}\left(i, W_t^i, a\right) \right|, \tag{3}$$

where N_E^{θ} is a sequential decision network parameterized by θ, a is one of the pre-defined value operations. $T_e = \min\left(t_m, t_e\right)$ is the ending step, where t_m is the maximum step, t_e is the step that chooses the ending operation. W_t^i is the weight vector that represents the chosen weights, which is initialized by a full-zero vector. W_t^i takes the form

$$W_{t+1}^i = \begin{cases} \{0, 0, 0, ...\} & if\ t = 0 \\ W_t^i \uplus a_t & otherwise \end{cases}, \tag{4}$$

where a_t is the operation chosen at the step t, and $\uplus$ is a weight updating operator (see also Eq. (7)). In step T the count V_T^i of the patch i takes the form

$$V_T^i = \sum_{t=0}^{T} \arg\max_{a} N_E^{\theta}\left(i, W_t^i, a\right) = \sum_{t=0}^{T} w_t^i, \tag{5}$$

where w_t^i forms W_t^i such that

$$W_t^i = \left(w_0^i, w_1^i, ..., w_{t-1}^i, 0, ...\right) . \tag{6}$$

Overall, the working flow of this weighing task is akin to scale weighing. In each step, the network N_E^θ (scale) evaluates the value difference between the image patch i and the value associated with the weight vector W_t^i (weights); according to the output of the network (needle), the agent chooses an action (add or remove a weight) to adjust V_T^i to approximate the ground-truth patch count $G\left(i\right)$ until they are equal (the scale is balanced). We present more details in the sequel.

2.2 Crowd Counting as Sequential Scale Weighing

We implement Eq. (3) within the framework of Q-leaning [29]. The elements of Q-learning include state, action, reward and Q value. They correspond to the scale pan, weights, designed rewards and needle in scale weighing.

State (Scale Pan). The state depicts the status of 'two scale pans'—the weight vector W_t^i and the image feature FV_I^i. Formally, the state $s = \{FV_I^i, W_t^i\}$.

According to [19], the data distribution is often long-tailed in crowd counting datasets with imbalanced samples. Liu *et al.* [19] shows that this issue could be alleviated by quantizing local counts and treating the count intervals as the learning target. We follow this idea to check the balancing condition of the scale.

Action (Weights). In Q-learning, an action is defined to modify the state. Since FV_I^i is fixed in s once it is extracted, the action is designed to only change W_t^i. We design an action pool in a way similar to the scale weighing system and the money system [42], i.e., $a = \{-10, -5, -2, -1, +1, +2, +5, +10, end\}$. It includes 8 value operations and one ending operation (indicating the scale is balanced). Given a new action a_t, W_t^i is modified by an updating operator $\uplus$

$$W_t^i \uplus a_t = \{w_0^i, ..., w_{t-1}^i, 0, 0, ...\} \uplus a_t = \{w_0^i, ..., w_{t-1}^i, a_t, 0, ...\} . \tag{7}$$

W_t^i records what weights are placed/removed from the scale pan before step $t-1$.

Reward Function. A reward scores the value of each action. We define two types of reward: ending reward and intermediate reward. In particular, we use a conventional *ending reward* and further design three counting-specific rewards—*force ending reward*, *guiding reward*, and *squeezing reward*.

Ending Reward. Following [2], we employ a conventional *ending reward* to evaluate the value of the 'end' action, defined by

$$R_e\left(E_{t_e-1}\right) = \begin{cases} +\eta_e & \text{if } |E_{t_e-1}| \leq \epsilon_1 \\ -\eta_e & \text{otherwise} \end{cases} , \tag{8}$$

where t_e is the step that the agent chooses the 'end' action, E_{t_e-1} is the absolute value error between the ground-truth count $G(i)$ and the accumulated value $V_{t_e-1}^i$, and ϵ_1 is the error tolerance. Here η_e=5, and ϵ_1=0.

Algorithm 1 Training Procedure of LibraNet

Initialize a *Buffer* $\leftarrow$ [ ], the Q-network N_Q^θ, and the backbone network N_b
for epoch $\leftarrow$ 0 to NumEpochs **do**
  Update the Q-network $N_Q^{\bar{\theta}} \leftarrow N_Q^\theta$
  for all image I in the training dataset **do**
    Compute the image feature $FV_I \leftarrow N_b(I)$
    for all patch i in image I **do**
      Initialize $W_0^i \Leftarrow \{0, 0, ...\}$
      Fetch the ground-truth patch count $G(i)$
      for $t \leftarrow 0$ to T_e **do**
        Obtain the state $s_t \leftarrow \{FV_I^i, W_t^i\}$
        Compute the Q value $Q_t \leftarrow N_Q^\theta(s_t)$
        Choose an action a_t with ϵ-greedy policy
        Compute the reward r according to Sec. 2.2
        Update W_{t+1}^i per Eq. (4)
        Obtain the next state $s_{t+1} \leftarrow \{FV_I^i, W_{t+1}^i\}$
        Buffer $\leftarrow (s_t, a_t, s_{t+1}, r)$
      end for
    end for
    Sample a batch B from the *Buffer* to train N_Q^θ per Eq. (16)
  end for
end for

Considering that the agent is hard to choose the 'end' action because of huge searching space, the agent is forced to stop when it exceeds the maximum step allowed. This is described by the *force ending reward*

$$R_{fe}(E_{t_m}) = \begin{cases} +\eta_e & \text{if } |E_{t_m}| \leq \epsilon_1 \\ -\eta_e & \text{otherwise} \end{cases}, \tag{9}$$

where E_{t_m} is the absolute value error at the maximum step t_m.

Intermediate Reward. In previous works [2, 14] that employ deep RL to deal with object localization, an intermediate reward is simply given according to the change of IoU. In counting, an optimal action can be computed to reach the balancing state faster. We thus introduce a *guiding reward* to push the agent to choose the optimal action, defined by

$$R_g(E_t, E_{t-1}, a_t, a_t^g) = \begin{cases} \eta_g & \text{if } a_t = a_t^g \\ \eta_+ & \text{if } E_t < E_{t-1} \\ \eta_- & \text{otherwise} \end{cases}, \tag{10}$$

where a_t is the action chosen in the step t, and a_t^g is the optimal action, given by

$$a_t^g = \arg\min_a \left| G(i) - \left(V_{t-1}^i + a\right) \right|. \tag{11}$$

In our implementation, η_g=+3, η_+=+1, and η_-=−1.

In our experiments, we find that, at the first several training epochs, the agent tends to choose large value operators that lead to overestimation. A possible

explanation is that, because of the huge searching space, the agent cannot search for actions smoothly. To reach the balancing state faster, we propose a *squeezing reward* to constrain the estimated value, defined by

$$R_s = \begin{cases} R_g\left(E_t, E_{t-1}, a_t, a_g\right) & \text{if } S\left(V_t^i, G\left(i\right)\right) = 1 \\ R_{sg}\left(E_t, E_{t-1}, a_t, a_g\right) & \text{otherwise} \end{cases}, \tag{12}$$

where R_g is the *guiding reward* (Eq. (10)). $S\left(V_t^i, G\left(i\right)\right)$ decides whether V_t^i is out of the tolerance range as

$$S\left(V_t^i, G\left(i\right)\right) = sign\left(G\left(i\right) \times \epsilon_2 - \left(V_t^i - G\left(i\right)\right)\right), \tag{13}$$

where ϵ_2 is a tolerance range set to 0.5 in this paper. If $S\left(V_t^i, G\left(i\right)\right)$=−1, we leverage a *squeezed guiding reward* to squeeze the estimation within the tolerance range, defined by

$$R_{sg}\left(E_t, E_{t-1}, a_t, a_t^g\right) = \begin{cases} \eta_{sg} & \text{if } a_t = a_t^g \\ \eta_s & \text{otherwise} \end{cases}, \tag{14}$$

where η_{sg}=−1, and η_s=−3. Notice that, in this reward function, all rewards are set to be negative such that the agent is encouraged to avoid choosing an action sequence that leads to overestimation.

Q Values (Needle). In Q learning, the Q value of an action is an estimation of the accumulated reward after this action is taken, which takes the form

$$Q\left(s_t, a_t\right) = \begin{cases} r & \text{if } a_t = \text{'end' or } t = t_m \\ r + \gamma \max_{a'} Q\left(s_{t+1}, a'\right) & \text{otherwise} \end{cases}, \tag{15}$$

where r is the reward coming from either R_e, R_{fe}, R_g or R_{sg}, the next state s_{t+1} is acquired after the action a_t is taken at the present state s_t, and γ is the reward discount factor set to 0.9 in our experiments. The Q value of each action is the output of DQN. It guides action selection and implies how the agent judges the scale balance. Hence Q value can be seen as the 'needle' of the 'counting scale'.

2.3 LibraNet

Here we give an overview of LibraNet (Fig. 2). LibraNet consists of two parts: a feature extraction backbone and a DQN. The backbone includes 5 convolutional blocks of VGG16 [38]. It aims to extract the feature map FV_I of an image I. Each spatial feature vector FV_I^i in FV_I and its weight vector W_t^i correspond to a 32×32 block in the original image. The backbone uses the model trained by [19] and is then fixed when training the DQN.

The core of LibraNet is the DQN. Its input is FV_I^i and W_t^i. In each step of the training stage, FV_I^i and W_t^i are concatenated and sent to a two-layer multi-layer perception (MLP) with 1024-dimensional hidden units in each layer, and the DQN outputs a 9-dimensional Q value Q_t. An action a_t chosen by ϵ-greedy policy

(Sec. 2.4) is then concatenated with W_t^i to obtain W_{t+1}^i (Eq. (4)). The estimation repeats until the 'end' action is reached or exceeds t_m steps. The output of DQN is the weighing vector $W_{T_e}^i$ for each patch i. When the weighing task terminates, $V_{T_e}^i$ is computed according to Eq. (5).

In the inferring stage, the agent chooses the action with the maximal Q value to obtain the weighing vector $W_{T_e}^i$ and the weighing value $V_{T_e}^i$ of each patch. Notice that $V_{T_e}^i$ is still the quantized count interval. It needs to be further mapped to a counting value with a class-count look-up table [19]. Finally we can sum all patch counts to obtain the image count.

2.4 Implementation Details

Following [29], we use a replay memory buffer [18] to remove correlations in the weighing process. We follow the standard DQN [29] structure which has a Q-network and a target network. The target network computing the target Q value ($\max_{a'} Q\left(s_{t+1}, a'\right)+r$) is fixed when training the Q-network, and we update the target network at the beginning of each epoch with the parameters of the Q-network. ℓ_1 loss is used for optimization. The overall loss is defined by

$$\ell = \sum_{(s_t, a_t, s_{t+1}, r) \in U(B)} \left| r + \gamma \max_{a'} N_Q^{\bar{\theta}}\left(s_{t+1}, a'\right) - N_Q^{\theta}\left(s_t, a_t\right) \right| / N \,, \tag{16}$$

where N_Q is LibraNet, θ and $\bar{\theta}$ are the parameters of the Q-network and the target network, respectively, r is the reward, and γ is the discount factor.

During training, we follow the ϵ-greedy policy: a random action is chosen either with a probability of ϵ or according to the maximum Q value. ϵ starts from 1 and decreases to 0.1 with a step of 0.05. To reduce computation cost, we update the model when every 100 samples are sent to the buffer. Considering that, the maximum quantized count interval is less than 80, the maximum step t_m is set to 8 (the maximum value operation is +10). Algorithm 1 summarizes the training flow. We use SGD with a constant learning rate of $1e^{-5}$.

Following [17], we crop 9 $\frac{1}{2}$-resolution patches. These patches are mirrored to double the training set. For the UCF-QNRF dataset [12], we follow BL [26] to limit the shorter side of the image to be less than 2048 pixels and to crop 512×512 patches for training.

3 Experimental Results

Here we validate the effectiveness of LibraNet, visualize the weighing process, compare it against other state-of-the-art methods, demonstrate its cross-dataset generalization, justify each design choice, and show its generality as a plug-in. We report the mean absolute error (MAE) and (root) mean square error (MSE).

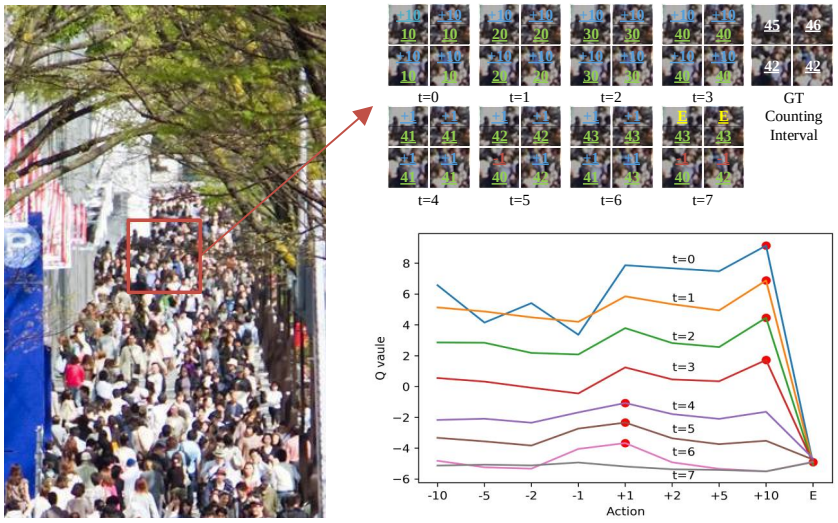

Fig. 3: Visualization of the inferring process of LibraNet. (upper right) Visualizations of action selection. We estimate the count interval for each 32×32 patch of the image. The weighing process is shown from t=0 to t=7, and the ground truth count intervals are shown in the right. For each patch, the lower green number is the accumulated value (the count interval), and the upper number is the value operator, including the value-increased operator (blue), the value-decreased operator (dull-red), and the ending operator 'E' (yellow). (bottom right) Estimated Q values in each step of the upper left patch. The red point in each step is the Q value of the chosen action.

3.1 Visualization of the Weighing Process

To understand how LibraNet works, we visualize the inferring process of one sample in Fig. 3. It can be seen that, in the first several steps, LibraNet tends to choose the action such that the estimation increases rapidly to approximate the ground truth. This is consistent with the target of *guiding reward* (Eq. (10)). When the accumulated value is close to the ground truth, LibraNet begins to choose actions with small values. This is similar to how we weight a stuff using a scale. Once the accumulated value equals to the ground truth, the weighing process terminates. Notice that, even if the maximum step is reached, LibraNet still produces a relatively accurate estimation due to *force ending reward* (Eq. (9)). Interestingly, we find that the agent chooses positive actions more frequently than negative ones, because i) the initial value is 0, and the target count is either 0 or positive. Thus, the agent tends to choose positive actions to approximate the ground truth, and ii) we design a squeeze guide reward (Eq. (14)) to avoid overestimation. This reward penalizes overestimation and further decreases the frequency of selecting negative actions.

To further analyze why the agent chooses certain actions, we visualize Q values of the top left patch. The ground truth count interval is 45, and the

Table 1: Comparison with state-of-the-art approaches on three crowd counting benchmarks. The lowest errors are boldfaced

Method	SHT Part_A		SHT Part_B		UCF_QNRF		UCF_CC_50	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
DRSAN [21]	69.3	96.4	11.1	18.2	—	—	219.2	250.2
CSRNet [17]	68.2	115.0	10.6	16.0	—	—	266.1	397.5
TEDnet [13]	64.2	109.1	8.2	12.8	113	188	249.4	354.5
SPN+L2SM [49]	64.2	98.4	7.2	11.1	104.7	173.6	188.4	315.3
BCNet [19]	62.8	102.0	8.6	16.4	118	192	239.6	322.2
BL [26]	62.8	101.8	7.7	12.7	88.7	154.8	229.3	308.2
CAN [22]	62.3	100.0	7.8	12.2	107	183	212.2	**243.7**
MBTTBF [39]	60.2	94.1	8.0	15.5	97.5	165.2	233.1	300.9
PGCNet [50]	57.0	**86.0**	8.8	13.7	—	—	244.6	361.2
S-DCNet [48]	58.3	95.0	**6.7**	**10.7**	104.4	176.1	204.2	301.3
LibraNet	**55.9**	97.1	7.3	11.3	**88.1**	**143.7**	**181.2**	262.2

agent chooses four consecutive +10, three +1 and one *End* actions. The final estimated interval is 43. In the first 4 steps, Q values excluding *End* are greater than 0 and have a clear distinction. It means that the agent is confident with its action selection. After four steps, the accumulated value is 40, which closes to the ground truth. In the last 4 step, Q values are less than 0, and the differences between each action is small, which implies the agent is aware of the closeness to the ground truth. To avoid overestimation, the agent becomes cautious to avoid a significantly wrong decision. Even if the final weighing value does not strictly equal to the ground truth, the estimation is not likely to shift away from the ground truth significantly. We can see that LibraNet follows exactly how a scale weighs a stuff, which means LibraNet indeed learns what we expect it to learn.

3.2 Comparison with State of the Art

We evaluate our method on three public crowd counting benchmarks: ShanghaiTech, UCF_CC_50 and UCF-QNRF.

The ShanghaiTech (SHT) Dataset [52] includes $1,198$ crowd images with $330,165$ head annotations. It has two parts: part A includes 482 images with varying resolution collected from Internet; part B includes 716 images of the same resolution collected from street surveillance videos. In part A, 300 images are used for training, and other 182 images for testing. Part B adopts 400 images for training and 316 images for testing. Results are shown in Table 1. We compare our method against other 10 state-of-the-art methods and report the best MAE in part A and comparable performance on part B.

The UCF_CC_50 Dataset [11] is a challenging crowd counting dataset with only 50 images. By contrast, there are $63,705$ people annotations, so the scenes are extremely congested. We employ 5-fold cross-validation when reporting the results and also compare LibraNet with other state-of-the-art approaches. The results shown in Table 1 verify that LibraNet outperforms other competitors and reports the best performance in MAE.

The UCF-QNRF Dataset [12] is a recent high-solution crowd counting dataset, which includes $1,535$ images with $1,251,642$ annotations. The images are officially split into two parts: 1201 images for training and 334 for testing. We compare

Table 2: Cross-dataset evaluations on the SHT (A and B) and UCF-QNRF (QNRF) datasets

Method	A → B		A→QNRF		B→A		B→QNRF		QNRF→A		QNRF→B	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
MCNN [52]	85.2	142.3	—	—	221.4	357.8	—	—	—	—	—	—
D-ConvNet [36]	49.1	99.2	—	—	140.4	226.1	—	—	—	—	—	—
SPN+L2SM [49]	21.2	38.7	227.2	405.2	126.8	203.9	—	—	73.4	119.4	—	—
BCNet [19]	20.5	37.9	131.9	230.6	138.6	230.0	240.0	419.6	71.3	123.7	16.1	26.1
BL [26]	—	—	—	—	—	—	—	—	69.8	123.8	15.3	26.5
LibraNet	**11.9**	**20.7**	**127.9**	**204.9**	**98.3**	**167.9**	**224.2**	**405.3**	**67.0**	**109.2**	**11.9**	**22.0**

Table 3: Ablation study on the SHT Part A dataset

Method	MAE	MSE
BCNet [19]	62.8	102.0
Imitation Learning [10]	64.7	102.8
W/O Guiding	149.8	261.3
W/O Force Ending	62.7	104.3
W/O Squeezing	63.5	102.7
Full Designs	**55.9**	**97.1**

Table 4: GAME on the SHT Part A dataset

	GAME0	GAME1	GAME2	GAME3
BCNet [19]	62.8	73.3	87.0	116.7
LibraNet	**55.9**	**68.0**	**82.1**	**113.1**

LibraNet with 7 recent methods. The results in Table 1 illustrate our method outperforms state-of-the-art methods in both MAE and MSE.

3.3 Cross-Dataset Generalization

To demonstrate the generalization of LibraNet, we conduct cross-dataset experiments by training the model on one dataset but testing on the other one. Results are shown in Table 2. LibraNet shows consistently better generalization performance than other competitors across all transfer settings.

3.4 Ablation Study

Here we validate basic design choices of LibraNet on the SHT Part_A dataset [52]. The results are shown in Table 3.

Local Accuracy. BCNet is the blockwise classification network proposed by [19]. This is our direct baseline, because LibraNet uses the backbone pretrained by [19]. Besides the image-level error, we also report the Grid Average Mean absolute Error (GAME) [9] in Table 4. GAME assesses patch-level counting accuracy. LibraNet outperforms BCNet in all GAME metrics, which suggests that LibraNet generates more locally accurate patch counts than BCNet. We believe this may be the reason why LibraNet significantly reduces the image-level error.

Optimal Action. In Sec. 2.2, we compute the optimal action to reach the balancing state faster. *Is it sufficient to learn a weighing model that only chooses the*

Table 5: Sensitivity analysis of the maximum step on the SHT Part A dataset

Step	4	6	8	10	12	14	16
MAE	126.3	59.0	55.9	57.7	62.5	60.7	56.9
MSE	243.0	106.6	97.1	99.2	101.3	100.5	93.9

Table 6: Sensitivity analysis of the tolerance range on the SHT Part A dataset

Range	0.1	0.3	0.5	0.7	0.9
MAE	61.0	59.7	**55.9**	60.1	59.9
MSE	104.5	100.1	97.1	**96.8**	103.3

Table 7: LibraNet as a plug-in

Method	MAE	MSE
ImageNet Regression	156.2	259.9
ImageNet Classification	140.4	230.3
ImageNet Regression+LibraNet	126.6	211.1
ImageNet Classification+LibraNet	119.7	203.4
TasselNetv2† [47]	68.6	110.2
TasselNetv2†+LibraNet	64.7	100.6
Blockwise Classification [19]	62.8	102.0
Blockwise Classification+LibraNet	**55.9**	**97.1**

optimal action? To justify this, we build another baseline 'Imitation Learning' [10] with the following optimization target

$$\max_{\theta} \sum_{i \in I} \sum_{t=0}^{T_e} \sum_{a=0}^{A_N} [a = a_{i,t}^{g}] \log \left(N_M^{\theta} \left(i, W_t^i, a \right) \right) , \tag{17}$$

where $a_{i,t}^{g}$ is the optimal action (Eq. (11)) of time t in patch i, A_N is the number of pre-defined action, N_M^{θ} is a sequential decision network, and $N_M^{\theta}\left(i, W_t^i, a\right)$ computes the probability of a-th action in i-th patch. In each step, N_M^{θ} selects the action with the maximum probability. Results in Table 3 show that *learning with only the optimal action is insufficient.*

Designed Rewards. From the 3-th to the 5-th rows of Table 3, we present the ablative studies on modified rewards. 'W/O Guiding' means training LibraNet without the 'guiding reward' (Eq. (10)) which simply sets $+1$ to error-decreased action and -1 to error-increased action, 'W/O Force Ending' means training LibraNet without the 'force ending reward' (Eq. (9)), and 'W/O Squeezing' means training LibraNet without the 'squeezing reward' (Eq. (14)). It is clear that all designed rewards benefit counting.

Parameters Sensitivity. To analyze the impact of the maximum action step t_m, we train LibraNet with t_m ranging from 4 to 16 on the SHT Part A dataset. Results are shown in Table 5. When t_m is not sufficient, LibraNet works poorly, because LibraNet cannot reach the neighborhood of ground truth even if the maximum value operation can be chosen in each step. We set $t_m = 8$ in all other experiments. We also evaluate the effect of the tolerance range (ϵ_2) in Eq. (12). Results are shown in Table 6. We observe that, LibraNet is not sensitive to this parameter, and the best result is achieved when $\epsilon_2 = 0.5$ on the SHT Part A dataset. We thus fix $\epsilon_2 = 0.5$. Furthermore, we analyze the effect of randomness.

Following [41], we run LibraNet for 6 times on the SHT Part A with different random seeds. The MAE is 56.4 ± 1.8, and MSE is 97.8 ± 2.3, which suggests LibraNet is not sensitive to randomness.

Execution Speed. Finally, we report the speed of LibraNet on a platform with RTX 2060 6 GB GPU and Intel i7-9750H CPU. It takes 158 ms to process an 1080×720 image, including 142 ms on backbone and 16ms on LibraNet. The result illustrates that LibraNet only introduces negligible computation costs.

3.5 LibraNet as a Plug-in

To show that LibraNet is a general idea and the pretraining with [19] is not the only opinion, here we apply LibraNet as a plug-in to other counting/pretrained models. Results are shown in Table 7.

First we attach LibraNet to a regression baseline and a classification baseline with ImageNet-pretrained VGG16 [38]. The VGG16 is fixed and concatenated with a trainable $1 \times 1 \times C$ or a $1 \times 1 \times 1$ convolution kernel to classify counting intervals or to regress patch counts. By using LibraNet, we observe more than 10% relative improvements over the regression and classification baselines. In addition, it can be observed that 'ImageNet Regression/Classifiaction+LibraNet' exhibits significantly worse performance than other comparing approaches. This suggests that pretraining the feature extraction backbone is important for counting. Such results are consistent with a recent observation on visual question answering systems [4] that *CNN features contain little information relevant to counting* [40].

The second model is a regression-based blockwise counter—TasselNetv2† [47]. 'TasselNetv2†+LibraNet' means extracting the feature map by the backbone pretrained by TasselNetv2† and then sending them to DQN to estimate the count. To adapt to regression-based weighing where the count values is continuous, we modify the pre-defined action pool $a = \{ -5, -2, -1, -0.5, -0.2, -0.1, -0.05, -0.02, -0.01, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5 \}$. Results show that 'TasselNetv2†+LibraNet' outperforms TasselNetv2, which illustrates the idea of scale weighing is also effective for the regression-based counter.

4 Conclusion

In this work, we have introduced a novel sequential decision paradigm to tackle crowd counting, which is inspired by the behavior of human counting and scale weighing. We implement scale weighing using deep RL and present a new counting model LibraNet. Experiments verify the effectiveness of LibraNet and explain how it works. For future work, we plan to extend LibraNet to other regression tasks. We believe that scale weighing is a general idea that may not be limited to counting.

Acknowledgement. This work is supported by the Natural Science Foundation of China under Grant No. 61876211 and Grant No. U1913602. Part of this work was done when L. Liu was visiting The University of Adelaide.

References

1. Araslanov, N., Rothkopf, C.A., Roth, S.: Actor-critic instance segmentation. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8237–8246 (2019)
2. Caicedo, J.C., Lazebnik, S.: Active object localization with deep reinforcement learning. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2488–2496 (2015)
3. Chan, A.B., Liang, Z.S.J., Vasconcelos, N.: Privacy preserving crowd monitoring: Counting people without people models or tracking. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–7. IEEE (2008)
4. Chattopadhyay, P., Vedantam, R., Selvaraju, R.R., Batra, D., Parikh, D.: Counting everyday objects in everyday scenes. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1135–1144 (2017)
5. Chen, K., Loy, C.C., Gong, S., Xiang, T.: Feature mining for localised crowd counting. In: Proc. British Machine Vision Conference (BMVC). p. 3 (2012)
6. Cohen, J.P., Boucher, G., Glastonbury, C.A., Lo, H.Z., Bengio, Y.: Count-ception: Counting by fully convolutional redundant counting. In: Proc. IEEE International Conference on Computer Vision Workshops (ICCVW). pp. 18–26 (Oct 2017). https://doi.org/10.1109/ICCVW.2017.9
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 886–893 (2005)
8. Diuk, C., Cohen, A., Littman, M.L.: An object-oriented representation for efficient reinforcement learning. In: Proc. International Conference on Machine learning (ICML). pp. 240–247. ACM (2008)
9. Guerrero-Gómez-Olmedo, R., Torre-Jiménez, B., López-Sastre, R., Maldonado-Bascón, S., Oñoro-Rubio, D.: Extremely overlapping vehicle counting. In: Iberian Conference on Pattern Recognition and Image Analysis. pp. 423–431 (2015)
10. Hussein, A., Gaber, M.M., Elyan, E., Jayne, C.: Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR) **50**(2), 1–35 (2017)
11. Idrees, H., Saleemi, I., Seibert, C., Shah, M.: Multi-source multi-scale counting in extremely dense crowd images. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2547–2554 (2013)
12. Idrees, H., Tayyab, M., Athrey, K., Zhang, D., Al-Maadeed, S., Rajpoot, N., Shah, M.: Composition loss for counting, density map estimation and localization in dense crowds. In: Proc. European Conference on Computer Vision (ECCV). pp. 532–546 (2018)
13. Jiang, X., Xiao, Z., Zhang, B., Zhen, X., Cao, X., Doermann, D., Shao, L.: Crowd counting and density estimation by trellis encoder-decoder networks. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6133–6142 (2019)
14. Kong, X., Xin, B., Wang, Y., Hua, G.: Collaborative deep reinforcement learning for joint object search. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1695–1704 (2017)
15. Laradji, I.H., Rostamzadeh, N., Pinheiro, P.O., Vazquez, D., Schmidt, M.: Where are the blobs: Counting by localization with point supervision. In: Proc. European Conference on Computer Vision (ECCV). pp. 547–562 (2018)
16. Lempitsky, V., Zisserman, A.: Learning to count objects in images. In: Advances in Neural Information Processing Systems (NIPS). pp. 1324–1332 (2010)

17. Li, Y., Zhang, X., Chen, D.: CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1091–1100 (2018)
18. Lin, L.J.: Reinforcement learning for robots using neural networks. Tech. rep., Carnegie-Mellon Univ Pittsburgh PA School of Computer Science (1993)
19. Liu, L., Lu, H., Xiong, H., Xian, K., Cao, Z., Shen, C.: Counting objects by blockwise classification. IEEE Transactions on Circuits and Systems for Video Technology (2019)
20. Liu, L., Qiu, Z., Li, G., Liu, S., Ouyang, W., Lin, L.: Crowd counting with deep structured scale integration network. In: Proc. IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)
21. Liu, L., Wang, H., Li, G., Ouyang, W., Lin, L.: Crowd counting using deep recurrent spatial-aware network. In: Proc. International Joint Conference on Artificial Intelligence (IJCAI). pp. 849–855. AAAI Press (2018)
22. Liu, W., Salzmann, M., Fua, P.: Context-aware crowd counting. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5099–5108 (2019)
23. Lu, H., Cao, Z., Xiao, Y., Zhuang, B., Shen, C.: TasselNet: counting maize tassels in the wild via local counts regression network. Plant methods **13**(1), 79 (2017)
24. Lu, H., Dai, Y., Shen, C., Xu, S.: Indices matter: Learning to index for deep image matting. In: Proc. IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3266–3275 (2019)
25. Lu, H., Dai, Y., Shen, C., Xu, S.: Index networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
26. Ma, Z., Wei, X., Hong, X., Gong, Y.: Bayesian loss for crowd count estimation with point supervision. In: Proc. IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6142–6151 (2019)
27. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Proc. International Conference on Machine Learning (ICML). pp. 1928–1937 (2016)
28. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
29. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
30. OpenAI: Openai five. `https://blog.openai.com/openai-five/` (2018)
31. Riedmiller, M., Gabel, T., Hafner, R., Lange, S.: Reinforcement learning for robot soccer. Autonomous Robots **27**(1), 55–73 (2009)
32. Ryan, D., Denman, S., Fookes, C., Sridharan, S.: Crowd counting using multiple local features. In: 2009 Digital Image Computing: Techniques and Applications. pp. 81–88. IEEE (2009)
33. Sam, D.B., Surya, S., Babu, R.V.: Switching convolutional neural network for crowd counting. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
34. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
35. Shi, M., Yang, Z., Xu, C., Chen, Q.: Revisiting perspective information for efficient crowd counting. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7279–7288 (2019)

36. Shi, Z., Zhang, L., Liu, Y., Cao, X., Ye, Y., Cheng, M.M., Zheng, G.: Crowd counting with deep negative correlation learning. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5382–5390 (2018)
37. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484 (2016)
38. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
39. Sindagi, V.A., Patel, V.M.: Multi-level bottom-top and top-bottom feature fusion for crowd counting. In: Proc. IEEE/CVF International Conference on Computer Vision (ICCV). pp. 1002–1012 (2019)
40. Stahl, T., Pintea, S.L., van Gemert, J.C.: Divide and count: Generic object counting by image divisions. IEEE Transactions on Image Processing **28**(2), 1035–1044 (2018)
41. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Thirtieth AAAI conference on artificial intelligence (2016)
42. Van Hove, L.: Optimal denominations for coins and bank notes: in defense of the principle of least effort. Journal of Money, Credit and Banking pp. 1015–1021 (2001)
43. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature pp. 1–5 (2019)
44. Wan, J., Luo, W., Wu, B., Chan, A.B., Liu, W.: Residual regression with semantic prior for crowd counting. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4036–4045 (2019)
45. Wang, C., Zhang, H., Yang, L., Liu, S., Cao, X.: Deep people counting in extremely dense crowds. In: Proc. ACM International Conference on Multimedia (ACMMM). pp. 1299–1302. ACM (2015)
46. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581 (2015)
47. Xiong, H., Cao, Z., Lu, H., Madec, S., Liu, L., Shen, C.: Tasselnetv2: in-field counting of wheat spikes with context-augmented local regression networks. Plant Methods **15**(1), 150 (2019)
48. Xiong, H., Lu, H., Liu, C., Liang, L., Cao, Z., Shen, C.: From open set to closed set: Counting objects by spatial divide-and-conquer. In: Proc. IEEE/CVF International Conference on Computer Vision (ICCV). pp. 8362–8371 (2019)
49. Xu, C., Qiu, K., Fu, J., Bai, S., Xu, Y., Bai, X.: Learn to scale: Generating multipolar normalized density maps for crowd counting. In: Proc. IEEE/CVF International Conference on Computer Vision (ICCV). pp. 8382–8390 (2019)
50. Yan, Z., Yuan, Y., Zuo, W., Tan, X., Wang, Y., Wen, S., Ding, E.: Perspective-guided convolution networks for crowd counting. In: Proc. IEEE/CVF International Conference on Computer Vision (ICCV). pp. 952–961 (2019)
51. Zhang, C., Li, H., Wang, X., Yang, X.: Cross-scene crowd counting via deep convolutional neural networks. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 833–841 (2015)
52. Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y.: Single-image crowd counting via multi-column convolutional neural network. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 589–597 (2016)