Abstract

In crowd counting, due to the problem of laborious labelling, it is perceived intractability of collecting a new large-scale dataset which has plentiful images with large diversity in density, scene, etc. Thus, for learning a general model, training with data from multiple different datasets might be a remedy and be of great value. In this paper, we resort to the multi-domain joint learning and propose a simple but effective Domain-specific Knowledge Propagating Network (DKPNet) for unbiasedly learning the knowledge from multiple diverse data domains at the same time. It is mainly achieved by proposing the novel Variational Attention (VA) technique for explicitly modeling the attention distributions for different domains. And as an extension to VA, Intrinsic Variational Attention (InVA) is proposed to handle the problems of over-lapped domains and sub-domains. Extensive experiments have been conducted to validate the superiority of our DKPNet over several popular datasets, including ShanghaiTech A/B, UCF-QNRF and NWPU.

1. Introduction

Crowd counting is a challenging problem since it suffers from multiple actual issues behind data distributions, such as high variability in scales, density, occlusions, perspective distortions, background scenarios, etc. A direct solution to mitigate these issues is to collect a large-scale dataset with abundant data variations like ImageNet[7], so as to encourage the learned model to be more robust and general. However, collecting such a large-scale dataset with rich diversity for crowd-counting training is intractable due to the difficulty in human-labeling. Specifically, in crowd counting, due to the limitation of various conditions, images collected by a research group might only contain certain types of variations and are limited in numbers. For example, as shown in Fig.1, one can observe that there are large variations in data distributions across different datasets. Images in ShanghaiTech A(SHA)[64] tend to show congested crowds, and those in UCF-QNRF(QNRF)[15] are more likely to depict highly-congested crowds and have more background scenarios, and those in NWPU[54] have much more diversities in scales, density, background, etc. In contrast, samples within ShanghaiTech B(SHB)[64] just prefer low density crowds and the ordinary street-based scenes.

However, in crowd counting, an interesting phenomenon can be observed when jointly training with multiple different datasets. As shown in Tab.1, if under the supervision of the 3-joint of SHA, SHB and QNRF, the deep model prefers to only improve the performances on SHA and QNRF, while sacrificing that on SHB (5% performance drop). Actually, this kind of phenomenon (i.e. biased/partial learning)

1Commonly, a domain often refers to a data set where samples follow the similar or same underlying data distribution[57].
exists widely in computer vision fields. It is because that deep models have the partial/biased learning behavior\cite{2}, i.e. deep models easily learn to focus on surface statistical regularities rather than more general abstract concepts. In other words, deep models will selectively learn the dominant data knowledge from certain dominant domains while ignoring other potential helpful information from the rest domains. To this end, developing an effective algorithm that can successfully utilize all the knowledge from different datasets remains important.

In this paper, we propose the **Domain-specific Knowledge Propagating Network** (DKPNet) for multi-domain joint learning, which intends to refine the propagated knowledge according to the domain-specific distributions and highlight all domains without bias. Specifically, a novel **Variational Attention** (VA) technique is introduced for facilitating the domain-specific attention learning. Based on VA, the output attention distribution can be easily controlled by the latent variable. And we apply the Gaussian Mixture distribution as the prior in the proposed VA for multi-domain learning. Furthermore, as an extension to VA, the **Intrinsic Variational Attention** (InVA) is proposed for handling the potential problems of overlapped-domains and sub-domains. VA and InVA both insist to provide domain-specific guidance for knowledge propagating, but start from coarse and intrinsic perspectives, respectively. In summary, the contributions of this paper are listed as follows:

- DKPNet is proposed to learn a general and robust density estimating model for crowd counting by multi-domain joint learning, which can successfully prevent the model from just learning several dominant domains, and can consistently improve the performances over all the datasets.
- VA/InVA are introduced to provide the domain-specific guidance for refining the propagating knowledge with the help of latent variable. To our knowledge, it is the first work to use variational learning in attention for crowd counting.
- Extensive experiments have been conducted on several popular datasets, including ShanghaiTech A/B\cite{64}, UCF-QNRF\cite{15} and NWPU\cite{54}, and achieve the state-of-the-art performances on the MAE evaluation.

### 2. Related Work

**Crowd Counting:** We review the recent works by the techniques they applied. Such techniques include multi-scale \cite{64, 42, 47, 30, 43}, multi-task \cite{63, 14, 27}, attention \cite{25, 48, 61, 62, 16, 35}, perspective map \cite{44, 59, 60}, GNN \cite{31}, loss functions \cite{32}\cite{52}, classification \cite{58} detection \cite{22, 41}, NAS \cite{13} and others \cite{47, 46, 24, 38, 26, 51, 55, 49, 45, 28, 1}. However, none of these works pay attention to the multi-domain learning in crowd counting.

**Cross-domain Learning in Crowd Counting:** Cross-domain learning can be categorized into one/few shot learning \cite{11, 40}, domain adaption \cite{20, 55, 8}, etc. \cite{40} presents a meta-learning inspired approach to solve the few-shot scene adaptive crowd counting problem, and \cite{11} further introduces one-shot scene-specific crowd counting. For domain adaption, CODA \cite{20} performs adversarial training with pyramid patches from both source- and target-domain, so as to tackle different object scales and density distributions. Wang et al. \cite{55} release a large synthetic dataset (GCC), and propose SE Cycle GAN to bridge the domain gap between the synthetic and real data. Gao et al. \cite{8} propose Multi-level Feature aware Adaptation (MFA) and Structured Density map Alignment (SDA) to extract domain invariant features and produce density maps with a reasonable distribution on the real domain.

**Multi-domain Learning:** Multi-domain learning aims at improving the performance over multiple domains. And it has been exploited in many fields \cite{36, 57, 39, 9, 29, 56, 19, 5}. Despite of heavy research on multi-domain learning in these fields, there are few corresponding research works in crowd counting. The most relevant work to our method is \cite{34}, \cite{34} presents domain-specific branches embedded behind a fixed ImageNet \cite{7} classification network. And, a domain classifier is proposed to decide which branch to process the input image. As a result, the final performance is not good and limited by the hard assignment of branches. Moreover, the computation cost of this work is in linearly correlation with the number of domains. However, different from this work, DKPNet is light in parameters and computation, and is more flexible and general in both training and testing phases, leading to much better performances.

**Variational Learning:** VAE\cite{18} has been widely explored and used in generative model families and is good at controlling the output distribution via the latent variable. Based on VAE, conditional-VAE\cite{50} proposes a deep conditional model for structured output prediction using Gaussian latent variables; $\beta$-VAE\cite{10} is proposed to use balancing term $\beta$ to control the capacity and the independence prior; $\beta$-TCVAE\cite{6} further extends $\beta$-VAE by introducing a total correlation term. All these methods aim at using variational learning for generating visually-good images. However, in this paper, we integrate the variational learning into the attention mechanism, and propose the Variational Attention for learning domain-specific attentions.

### Table 1: MAE Results of IT/JT in 3-Joint datasets.

<table>
<thead>
<tr>
<th>Methods</th>
<th>SHA</th>
<th>SHB</th>
<th>QNRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual Training (IT)</td>
<td>60.6</td>
<td>8.8</td>
<td>97.7</td>
</tr>
<tr>
<td>Joint Training (JT)</td>
<td>60.2 (↓)</td>
<td>9.3 (↑)</td>
<td>92.8 (↓)</td>
</tr>
</tbody>
</table>
3. Proposed Method

In this section, we will first give the motivation of our method in Sec.3.1, then introduce the Variational Attention (VA) and the Intrinsic Variational Attention (InVA) modules in Sec.3.2 and Sec.3.3, resp. Finally provide the whole pipeline of Domain-specific Knowledge Propagating Network (DKPNet) in Sec.3.4.

3.1. Motivation

As experimented in Tab.1, optimizing a deep model by directly employing all the data coming from SHA, SHB and QNRF datasets gives rise to the problem of biased domain learning behavior[2, 4], i.e. the deep model prefers to mainly focusing on the learning of the dominant domains instead of all the domains. This will lead to confusions in model prediction when giving data from non-dominant domains. Therefore, in multi-domain cases, data distributions will be perturbed, resulting in difficulty in intrinsic attention outputs for different domains, and when these outputs are from different-domain attention distributions, we maximizing the log(\(p_\theta(y|x,l)\)) as follows:

\[
\log(p_\theta(y|x,l)) = \log(\int \frac{p_\theta(y,z|x,l)}{q_\phi(z|x,l)} q_\phi(z|x,l)dz)
\]

satisfy

\[
\geq E_{q_\phi(z|x,l)}\log\left(\frac{p_\theta(y,z,x,l)p_\phi(z|x,l)}{q_\phi(z|x,l)}\right)
\]

\[
= E_{q_\phi(z|x,l)}\log(p_\theta(y|z,x,l)) - KL(q_\phi(z|x,l)||p_\phi(z|x,l))
\]

This objective function is the evidence lower bound (ELBO) and includes two terms. The first term tries to maximize the likelihood to improve the confidence of prediction, in other words, it tries to produce good attention proposals so as to benefit the density estimation (in this paper, it corresponds to density estimating loss, which will be described later), \(\int q_\phi(z|x,l)||Y(y) - \hat{Y}(y)||^2dz\), where \(\hat{Y}(y)\) is the ground-truth of density map and \(Y(y)\) is estimating result based on the attention output \(y\). The second term refers to the KL divergence between the variational distribution \(q_\phi(z|x,l)\) (parameterized by \(\phi\)) and the prior distribution \(p_\phi(z|x,l)\), as it is prior distribution we use \(p_\phi(z|x,l) = p_\phi(z|x,l)\). Since the output attention distributions for different domains should be different with each other, we set the prior distribution of the latent variable \(z \in \mathbb{R}^c\) to a commonly used Gaussian mixture distribution with \(C\) Gaussian components, where \(C\) is the number of domains and \(d\) is the dimension of \(z\):

\[
z \sim \sum_{c=0}^{C-1} \gamma_c N(u_c, \Sigma_c), \forall c, \gamma_c \geq 0, \sum_{c=0}^{C-1} \gamma_c = 1
\]

For each domain, \(u_c\) is the corresponding mean vector and \(\Sigma_c\) is the \(d\)-dimensional covariance matrix. For convenience, we set \(\gamma_c = \frac{1}{C}\) and \(\Sigma_c\) is diagonal matrix throughout this paper. Then, the second term in Eq. 1 can be expressed as:

\[
KL(q_\phi(z|x,l)||p_\phi(z|x,l)) = \frac{1}{2} \log\left(\frac{det(\Sigma_c)}{det(\Sigma_\phi)}\right) - d + tr(\Sigma_c^{-1}\Sigma_\phi) + (u_c - u_\phi)^T(\Sigma_c^{-1}(u_c - u_\phi))
\]

where \(l = c\), the parameterized distribution \(q_\phi(z|x,l) \sim N(u_\phi, \Sigma_\phi)\), and \(u_\phi, \Sigma_\phi\) are the outputs of model \(\phi\).

As the \(E_{q_\phi(z|x,l)}\log(p_\theta(y|z,x,l))\) is computationally intractable and the total process should be differentiable, we use the reparameterization trick [18] for computation as:

\[
E_{q_\phi(z|x,l)} \log(p_\theta(y|z,x,l)) \simeq \frac{1}{N} \sum_{j=1}^{N} \log(p_\theta(y|z^j,x,l))
\]

where \(z^j\) is sampled by \(u_\phi + \Sigma_\phi \odot \epsilon, \epsilon \sim \mathcal{N}(0, I)\).

As mentioned before, each domain is supposed to have a Gaussian distribution, however, it is hard to factiously set and fix the prior parameters \(u_c, \Sigma_c\). We propose to make
Finally, \( V_A \) can be optimized by minimizing the following function, here, we minimize its upper-bound function \( \log \) and the attention output \( y \) have different distributions. And for minimizing Eq.6 can help regularizing different domains to sum them to be the learnable parameters for adaptively adjusting and add a distribution regularizer for getting non-trivial solutions. Considering the semantic relations between domains, we propose to regularize the similarity between the solutions. Considering the semantic relations between doing and add a distribution regularizer for getting non-trivial them to be the learnable parameters for adaptively adjust-

\[
\begin{align*}
L &= \log(p(y|x, l)) + \frac{1}{N} \sum_{l} \sum_{j} \log(p(y|x, l)) + KL(q_\phi(z|x, l)||p(z|x, l)) \\
&\quad + \log \left( \sum_{i=0}^{C-1} e^{z_i^T u_c - z_i^T u_{c-1}} \right) + \log^2 \left( \det(\Sigma_c) \right)
\end{align*}
\]

and the attention output \( y \) will be further used to re-weight the input tensor \( x \) as \( x = x \odot y \).

\[ L_{\text{reg}} = \log \left( \sum_{i=0}^{C-1} e^{z_i^T u_c - z_i^T u_{c-1}} \right) \]

Remark: By introducing and modeling the latent variable, the output attention will be domain-related, such that the domain-specific knowledge can be well captured and learned in multi-domain cases. As shown in Fig. 2, different from SE attention, our VA can produce more separable attention distributions for different domains with the help of latent variable modeling. And the latent variable is assumed to be Gaussian Mixture distribution, such that for each domain, an independent Gaussian distribution can be applied and used to control the attention outputs.

3.3. Intrinsic Variational Attention

Actually, the above VA simply assumes that each dataset belongs to a single domain. However, it does not hold on in many cases and there might be two frequent problems: (1) domain-overlaps across different datasets and (2) sub-domains within the same dataset. Therefore, labels \( l \) will be too coarse to provide the fine-grained and exact guidance for domain-specific attention learning, still leaving behind some confusions in attention learning. To this end, in order to capture the intrinsic domain labels for accurate domain-specific attention learning, we extend our VA into \textbf{Intrinsic Variational Attention} (InVA). It is mainly achieved by using Clustering(CL) labels and \textbf{Sub-Gaussian Components(SGC)} which intend to mitigate the problems of domain-overlaps and sub-domains, respectively.

Specifically, in order to tackle the domain-overlaps, it is required to reassign the more correct domain-labels to the training data instead of using the original dataset-label \( l \). Thus, we first train a VA and then perform the Gaussian-Mixture clustering\(^3\) over the attention proposals outputed by the VA module. The number of clusters is set to \( \hat{C} \). After clustering, the new domain labels \( \hat{l} \in \{0, \ldots, \hat{C} - 1\} \), i.e. CL labels, are reassigned to the original training data.

Moreover, there also might be sub-domains within each clustered domain since the clustering is unsupervised and not capable of finding each potential sub-domains. In order to cope with the potential sub-domains, we propose to use SGC for the latent variable modeling. Specifically, we assume there are at most \( k \) sub-domains in each clustered domain. Thus, the latent variable \( z \in \mathbb{R}^d \) will turn to be a Sub-Gaussian Mixture distribution as follows:

\[
\begin{align*}
z &\sim \sum_{c=0}^{\hat{C}-1} \frac{1}{\hat{C}} N(u_c, \Sigma_c) \\
\text{where } u_c &= \arg\max_{u_{c,i}} \sigma(u_{c,1}^T u_\phi, \ldots, \sigma(u_{c,k}^T u_\phi))
\end{align*}
\]

\( \sigma \) means Dropout with drop-rate 0.2, \( u_{c,k} \) is center vector of the \( k \)-th sub-component in the \( c \)-th Gaussian. For simplicity, we use the same \( \Sigma_c \) for the sub-gaussian components. Finally, using the CL labels \( \hat{l} \) and substituting the

\(^3\)We also tested other clustering methods, e.g. Kmeans, DBSCAN, etc., the performances are similar.
Figure 3: The pipeline of the proposed DKPNet. It contains two-stage training: VA and InVA are used in Stage-I and Stage-II, respectively. Before Stage-II training, we will first get the CL labels from the attention outputs of VA by clustering, and then reassign the training images with these CL labels.

The latent variable $z$ in Eq. 8 into Eq. 7 can obtain the loss function $L_{InVA}$ for training InVA.

**Remark:** CL labels focus on handling the overlapped-domains across different datasets. SGC allows the existence of sub-domains in each clustered domain and is capable of adaptively optimizing these sub-domains. Thus, equipping with both CL labels and SGC, InVA module can provide more accurate guidance for knowledge refining by attention according to the “intrinsic” domains of data. As the basic module structure is similar with VA except for the CL labels and prior distribution of $z$, here for the limitation of paper length, we omit the figure for showing InVA.

### 3.4. DKPNet

Now, we will introduce the whole pipeline of our DKPNet as in Fig 3. Here, we take the truncated HRNet[53](for parameter lightness, we only use parameters from stage1 to stage3) as our backbone. During training, the mini-batched images are randomly sampled from all datasets and then are fed into the backbone together. A $1 \times 1$ convolution with 512 channels is applied after, producing a 4-D tensor $x$. Then tensor $x$ will be passed to VA/InVA for producing the domain-specific attention weights $y$, which will be applied on $x$ by channel-wise attention, resulting in a new tensor $x = x \circ y$. Notably, VA/InVA are used in stage-I and stage-II, respectively. And for each stage training, the backbone is initialized by the ImageNet-pretrained model. Finally, for predicting the density maps $\hat{Y}(y)$, three $1 \times 1$ convolution layers (with channels 64, 32, 1, respectively) are adopted. After replacing $-\log(p(y|z^j,x,l))$ by the density estimating loss $\|\hat{Y}(y)_i - Y_i\|^2$, the whole DKPNet can be optimized by the following objective function:

$$L = \frac{1}{2B} \sum_{i=1}^{B} \|\hat{Y}(y)_i - Y_i\|^2_2 + KL(q_{\phi}(z|x,l)||p(z|x,l))$$

$$+ \log\left(\sum_{i=0}^{C-1} e^{z_i^T w_i - z_i^T u_c}\right) + \log^2(det(\Sigma_c))$$

(9)

$B$ is the batch size and for VA/InVA, the latent variable $z$ are modeled by Eq. 2 and Eq. 8, resp.

**Remark:** VA/InVA are performed in order, and progressively aim at refining the propagating information flows by attention, such that the data from different distributions can be unbiasedly treated and learned, without inducing confusions in predictions.

### 4. Experiments

**Datasets:** We conduct experiments on ShanghaiTech A/B[64], UCF-QNRF[15] and NWPU[54]. SHA contains 482 crowd images with crowd numbers varying from 33 to 3139, where 300 images are used for training and the rest 182 images are used for testing. SHB contains 716 images with crowd numbers varying from 9 to 578, where 400 images are employed for training and the rest 316 images are for testing. QNRF [15] contains 1,535 images. These images are split into the training set with 1,201 images and the testing set with 334 images, respectively. This dataset has much more annotated heads and prefers highly-congested density. NWPU dataset[54] is a new public dataset which consists of 5,109 images, including 3,109 training images, 500 val images and 1,500 test images, where the test images can only be evaluated on the official website.

**Mention & Notations:** As in multi-domain joint learning, these datasets are trained simultaneously by a single model, and tested individually, i.e. reporting the results on each datasets separately. “DKPNet($c, k$)” denotes that we use $c$ clustered domains and at most $k$ sub-domains in the proposed DKPNet.

**Implementation details:** The proposed DKPNet is applied on the truncated HRNet-W40[53] architecture which is pretrained on ImageNet[7]. For model training, we adopt the Adam[17] optimizer with default betas = (0.9, 0.999), set the start learning rate to be 0.00005 for both the pretrained backbone and the new added layers, and use a total of 450 epochs with the learning rate decreased by a factor of 2.5 per 250 epochs. For data preprocessing, we adopt fixed Gaussian kernels with size 15 to generate the ground-truth density maps. For images with the shortest side smaller than 416, we will resize the shortest side to 416 by keeping the aspect ratio. Then during training, random-cropping with
Table 2: Results on SHA, SHB, QNRF and NWPU. “Individual” means the models are trained by only one individual dataset. “3-Joint” and “4-Joint” refer to using the joint dataset of (SHA,SHB,QNRF) and (SHA,SHB,QNRF,NWPU), respectively. For each joint dataset, only a single model is trained. Moreover, “IT” refers to training models just with the individual datasets. “JT” means merging all the datasets for training. For fair comparison, IT and JT are performed under the same training settings as our DKPNet. The best results are in red color. NWPU(V) and NWPU(T) indicate Val and Test sets on NWPU, resp.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
</tr>
<tr>
<td>CSRNet [21]</td>
<td>68.2</td>
<td>115.0</td>
<td>10.6</td>
<td>16.0</td>
<td>-</td>
</tr>
<tr>
<td>CANet [26]</td>
<td>62.3</td>
<td>100.0</td>
<td>7.8</td>
<td>12.2</td>
<td>107.0</td>
</tr>
<tr>
<td>SFCN [55]</td>
<td>67.8</td>
<td>107.5</td>
<td>7.6</td>
<td>13.0</td>
<td>102.0</td>
</tr>
<tr>
<td>DSSINet [23]</td>
<td>60.6</td>
<td>96.0</td>
<td>6.9</td>
<td>10.3</td>
<td>99.1</td>
</tr>
<tr>
<td>Bayes [32]</td>
<td>62.8</td>
<td>101.8</td>
<td>7.7</td>
<td>12.7</td>
<td>88.7</td>
</tr>
<tr>
<td>DM-Count [52]</td>
<td>59.7</td>
<td>95.7</td>
<td>7.4</td>
<td>11.8</td>
<td>85.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training Dataset: Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT (baseline1)</td>
</tr>
<tr>
<td>JT (baseline2)</td>
</tr>
<tr>
<td>MB [34]</td>
</tr>
<tr>
<td>DKPNet (c=5,k=2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training Dataset: 3-Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>JT (baseline2)</td>
</tr>
<tr>
<td>MB [34]</td>
</tr>
<tr>
<td>DKPNet (c=5,k=2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training Dataset: 4-Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>JT (baseline2)</td>
</tr>
<tr>
<td>MB [34]</td>
</tr>
<tr>
<td>DKPNet (c=5,k=2)</td>
</tr>
</tbody>
</table>

In order to highlight the significance of the proposed DKPNet, we compare it with some recent remarkable works over the popular challenging benchmarks, including ShanghaiTech A/B [64], UCF-QNRF [15] and NWPU [54]. IT and JT are conducted with the same training configurations as our DKPNet, and both of them are set as our baselines.

4.1. Comparison with State-of-the-Arts

In order to highlight the significance of the proposed DKPNet, we compare it with some recent remarkable works over the popular challenging benchmarks, including ShanghaiTech A/B [64], UCF-QNRF [15] and NWPU [54]. IT and JT are conducted with the same training configurations as our DKPNet, and both of them are set as our baselines.

More Data & More Better? To answer this question, we provide some results as in Tab.2. From this table, one can observe that merging more datasets for training a robust estimating model is infeasible. For example (1) in 3-joint case, comparing JT with IT, JT will biasedly sacrifice the performances on SHB, i.e. raising the MAE from 8.8 to 9.3 (the same phenomenon can also be observed in the 4-joint datasets training case, i.e. JT raises the MAE on SHB from 8.8 to 9.7). (2) comparing JT results in 3-Joint case with those in 4-Joint case, one can observe that the performances are similar in both 3-Joint and 4-Joint cases, and even the performance on SHB in 4-Joint case is worse than that in 3-Joint case. From the above observations, we can get conclusion that using more datasets cannot easily result in a much better model.

Effectiveness of DKPNet: To this end, DKPNet is proposed and applied in both 3-joint and 4-joint cases. As in Tab.2, one can observe that DKPNet can successfully use more data for learning a better model, i.e. it can consistently improve the performances over both IT and JT baselines. For example, (1) when using 3-Joint(or 4-Joint) datasets DKPNet can surpass the baseline IT by a large margin(e.g. obtaining 61.8 MAE on NWPU(V) with 24% gains); (2) when using 4-Joint datasets, DKPNet can further improve the performances over DKPNet in 3-Joint case. These results verify the importance of our DKPNet for propagating domain-specific knowledge by using the latent-variable-constrained attention. Moreover, DKPNet(c=5,k=2) outperforms all the listed methods in MAE evaluation by a large margin, demonstrating the effectiveness of our DKPNet for multi-domain learning in crowd counting. Notably, DKPNet requires only One model for all the evaluations, while other methods have to require the corresponding trained model for each dataset. And the visualizations of density maps of JT and DKPNet are in Fig.4.

Comparison with Multi-Branch Learning: Moreover, comparing with the most related work MB [34] which uses a shared backbone followed by multiple branches for processing the different datasets(in which the parameter number and computational cost will increase with the number of datasets in a linear manner), DKPNet can surpass it by a large margin, as shown in Tab. 2, with negligible increases of parameters and computational costs. And MB uses...
the hard assignment of branches which is too “arbitrary” and cannot handle the problems of overlapped-domains and sub-domains, leading to a few performance improvements over the baseline JT. In contrast, DKPNet is much softer for handling the different data domains by using latent variable constrained attention, and is capable of handling these above problems.

4.2. Component Analysis

Why the domain-specific attention works? In order to answer this question and to demonstrate the effectiveness of our enhanced attention method in DKPNet, we take the SE[12] attention for comparison. For fair comparison, we train the SE-based model via adopting the same training configurations and backbone as our DKPNet, only replacing the VA/InVA modules with the SE attention module. We first provide the comparisons of attention spaces via tsne visualization as in Fig.5. Specifically, as in the 3-Joint case, when training with the SE module, the attention distributions for different datasets are very close to each other and even confuse with each other, e.g. the attention weights for SHB are very close to those for SHA and QNRF, even over-laps with them. However, this confusing attention output does not bring the performance improvements on the SHB dataset (i.e. as shown in Tab.3, MAE result on SHB will be weakened from 8.8 to 9.4). It is because that the deep model will produce the accurate predictions if and only if it can exactly capture the true data distributions and then specifically learn knowledge from them. Therefore, when employing the SE attention which has no explicit domain guidance, the images in SHB (which have clear distribution differences with images in (SHA, QNRF)) will be wrongly assigned with the (SHA, QNRF)-like attention weights, resulting in further confusion in model prediction. Moreover, the same phenomenon can also be observed in the 4-Joint case, i.e. SE produces much more confused attention distributions, meanwhile it results in the performance drops on both SHB and QNRF, i.e. drop from 8.8 to 9.6 and from 97.7 to 97.8, resp.

However, in contrast, as shown in Fig.5 and Tab.3, when training with our domain-specific attention modules, the output attention spaces are much more separable than those outputted by SE. As a result, the propagating information can be handled specifically and exactly for producing correct predictions. Finally, the performances on all datasets can be consistently improved without biases, outperforming SE by a large margin. These phenomena demonstrate the necessity and importance of explicitly learning the domain-specific attention for multi-domain joint training.

Effect of the VA and InVA: As in Tab.4 and Fig.6, we conduct the quantitative and qualitative comparisons on DKPNet. For convenience, we will take the 3-Joint case for major description. Specifically, the VA tries to learn a coarsely domain-specific attention output via the latent variable $z$. This leads to the consistent performance improvements over both the baselines IT and JT (e.g. $57.5 \times 60.2/60.6, 7.6$ vs. $9.3/8.8$ and $87.9$ vs. $92.8/97.7$ on SHA, SHB and QNRF respectively.). The learned attention distributions for different datasets are relatively separable ex-

![Figure 5: Tsne visualizations of the attention outputs of SE and our DKPNet. Different colors refer to different domains based on the GT labels and CL labels.](image)

![Figure 4: Visualizations of test samples in the 4-Joint case. (a), (b), (c) and (d) are the input, density maps of JT, DKPNet(c=5,k=2) and Ground-Truth, resp.](image)

Table 3: MAE comparisons between SE[12] and DKPNet. For 3-Joint and 4-Joint cases, we use DKPNet(c=3,k=3) and DKPNet(c=5,k=2), respectively.

<table>
<thead>
<tr>
<th>Methods</th>
<th>SHA</th>
<th>SHB</th>
<th>QNRF</th>
<th>SHA</th>
<th>SHB</th>
<th>QNRF</th>
<th>NWPU(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT</td>
<td>60.6</td>
<td>9.0</td>
<td>97.7</td>
<td>60.6</td>
<td>8.8</td>
<td>97.7</td>
<td>81.7</td>
</tr>
<tr>
<td>JT</td>
<td>60.2</td>
<td>9.3</td>
<td>92.8</td>
<td>JT</td>
<td>59.9</td>
<td>9.7</td>
<td>91.1</td>
</tr>
<tr>
<td>VA</td>
<td>75.5</td>
<td>7.6</td>
<td>87.9</td>
<td>VA</td>
<td>57.6</td>
<td>7.2</td>
<td>87.6</td>
</tr>
<tr>
<td>InVA(c=3)</td>
<td>57.3</td>
<td>7.5</td>
<td>86.7</td>
<td>InVA(c=5)</td>
<td>56.7</td>
<td>7.7</td>
<td>84.7</td>
</tr>
<tr>
<td>InVA(c=3,k=3)</td>
<td>56.7</td>
<td>6.9</td>
<td>85.1</td>
<td>InVA(c=5,k=2)</td>
<td>55.6</td>
<td>6.6</td>
<td>81.4</td>
</tr>
</tbody>
</table>

![Table 4: The MAE results comparisons of VA/InVA.](image)

![Figure 6: Tsne visualizations of the attention outputs.](image)
except for some special cases (some overlapped distributions). Moreover, considering that the dataset-labels are not the exact definitions of the intrinsic data domains, we propose the InVA module to further explore the intrinsic domains by data clustering and fine-grained Gaussian Mixture distribution modeling. For example, when using InVA(c=3), one can observe that the aforementioned overlapped distributions can be properly handled to some extent, obtaining further consistent performance improvements over VA (e.g. improving from 57.5 to 57.3, from 7.6 to 7.5 and from 87.9 to 86.7 on SHA, SHB and QNRF, resp.). Furthermore, InVA(c=3,k=3) is proposed to cope with the potential sub-domains within each clustered domain. One can observe that after giving chances of learning the potential sub-domains, the attention distributions of each clustered domains are more compact than before and the quantitative performances are further improved (e.g. improving the MAE from 57.3 to 56.7, 7.5 to 6.9 and 86.7 to 85.2 on SHA, SHB and QNRF, resp.). And the similar performance improvements can also be observed in the 4-Joint case.

In summary, DKPNet concentrates on progressively learning the domain-specific guided information flow by the two-staged training framework, where VA and InVA are performed in order.

**Ablation Study on the value of \((c,k)\):** As mentioned before, in InVA, we will first obtain the clustered domains that are separable to some extent in the macro-perspective, and then handle the potential sub-domains. The experimental results are shown in Tab.5, one can observe that, for the 3-Joint case, \(c=3\) works the best. It is reasonable since from a global view of the attention space output by VA (see Fig.6), \(c=3\) can well separate the different distributions without much confusion. And for each clustered domain, \(k=2,3,4\) works better than \(k=1\) since the potential sub-domains can be explicitly learned, and we experimentally find \(k=3\) is the best. Moreover, for the 4-Joint case, we experimentally find \((c=5,k=2)\) works the best.

**Sub-domain analysis:** In order to explicitly show the learning results of the sub-domains, we compute the cosine similarities between sub-domain centers for the 3-Joint case. \(\text{Sim}(q,p)\) means the cosine similarity between sub-centers \(u_{c,q}\) and \(u_{c,p}\).

<table>
<thead>
<tr>
<th>Methods</th>
<th>SHA</th>
<th>SHB</th>
<th>QNRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>InVA(c=3)</td>
<td>57.3</td>
<td>7.5</td>
<td>86.7</td>
</tr>
<tr>
<td>InVA(c=4)</td>
<td>59.6</td>
<td>7.7</td>
<td>88.2</td>
</tr>
</tbody>
</table>

**Table 5: MAE results on the value of \((c,k)\) in the 3-Joint case.** For convenience, we will omit writing \(k\) when \(k=1\).

<table>
<thead>
<tr>
<th>Methods</th>
<th>SHA</th>
<th>SHB</th>
<th>QNRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>JT(baseline)</td>
<td>59.9</td>
<td>9.7</td>
<td>91.1</td>
</tr>
<tr>
<td>DKPNet</td>
<td>55.6</td>
<td>6.6</td>
<td>81.4</td>
</tr>
</tbody>
</table>

**Table 6: Cosine similarities between sub-domain centers for the 3-Joint case.**

<table>
<thead>
<tr>
<th>C0</th>
<th>C1-CL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sim(0,1)</td>
<td>0.61</td>
</tr>
<tr>
<td>Sim(0,2)</td>
<td>0.75</td>
</tr>
<tr>
<td>Sim(1,2)</td>
<td>0.67</td>
</tr>
<tr>
<td>avg-sim</td>
<td>0.68</td>
</tr>
</tbody>
</table>

Model size: From Tab.9, one can observe that the total parameter number of DKPNet is fewer than [32, 52], but DKPNet can surpass them by a large margin, showing that DKPNet is indeed “light and sweet”. Moreover, DKPNet only needs one single model for all data evaluations, while [32, 52] have to train many corresponding models for evaluations which are heavy and complicated.

5. Conclusion

In this paper, we propose DKPNet for learning the robust and general density estimating model for crowd counting by multi-domain joint learning. Specifically, DKPNet is a two-stage training framework, where the VA module is used in Stage-I for coarsely guiding the domain-specific attention learning, and the InVA module is employed in Stage-II for exploring the intrinsic domains by handling both the problems of overlapped-domains and sub-domains, so as to provide more accurate guidance for domain-specific attention learning. Finally, extensive experiments have been conducted on four popular benchmarks to validate the necessity and effectiveness of our method.

Acknowledgments: We hereby give specific thanks to Alibaba Group for their contribution to this paper.
References


Reference


