CDGNet: Class Distribution Guided Network for Human Parsing: Supplementary Material

Kunliang Liu[1,2], Ouk Choi[3], Jianming Wang[2*], and Wonjun Hwang[1*]
[1]Ajou University, Korea, [2]Tiangong University, China, [3]Incheon National University, Korea
tjpulkl@ajou.ac.kr, ouk.choi@inu.ac.kr, wangjianming@tiangong.edu.cn, wjhwang@ajou.ac.kr

1. Experimental Details

1.1. Class Horizontal and Vertical Distribution

We describe the details of how to achieve class horizontal and vertical distribution labels based on the Ground Truth for human parsing. We utilize LIP dataset as the example images for introducing our process.

One-hot Encoding: In the scene segmentation and human parsing task, for each image, there is one corresponding ground truth that assigns a semantic category integer number for each pixel. To explicitly consider each category distribution trait in 2D space, the direct technique is to calculate each category bounding box. However, bounding box only reflects the distribution range in horizontal and vertical dimensions. It does not represent the distribution density in the class bounding box. As a result, bounding box has limitation in the description of class distribution trait. In our paper we do not adopt bounding box as other methods [1–4] do. We intentionally calculate each class spatial distribution from one-hot encoding feature of paring ground truth. Given the ground truth L, we use the one-hot encoding mechanism to encode each categorical integer label in the ground truth to generate a 3D matrix M of size $N \times H \times W$, here N is category number, H and W are the height and width of ground truth, respectively.

Statistic of Distribution Trait: Given matrix M, we can count the number of 1 in horizontal and vertical directions. The larger of the statistic number means the greater of the distribution density. By this means, we can obtain each category horizontal and vertical distribution traits that contain more information than bounding box. To regularize each class distribution density in the range of 0 to 1, we divide each row's value by its max value. In LIP dataset, there are totally 20 classes including background. Therefore, we obtain 20 classes horizontal and vertical distribution maps. In most case, the background surrounds other classes and its distribution determines by all other categories together. So, we exclude background when utilizing class distribution. It is easy to implement by setting the corresponding row to 0, as pseudocode Algorithm 1 shows. The example of every class horizontal and vertical distributions in a single human image is as Fig. 1 shows. Each class horizontal and vertical distribution range and density are clearly expressed in the heatmaps, such as hair, coat and pant, etc. The distribution range and density of each class are even more obvious in Fig. 2. For instance, the hats that the two persons wear in the bottom of Fig. 2, there are two highlight areas in the vertical and horizontal heatmaps, respectively, correspond-

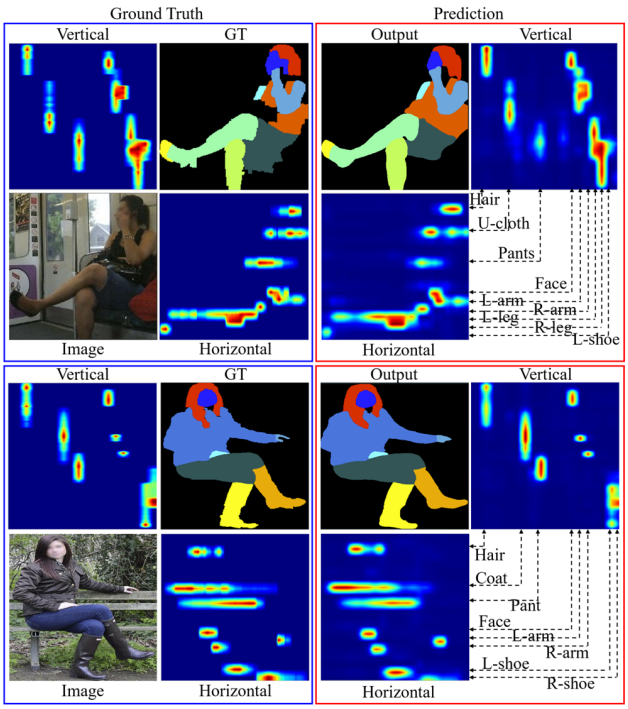

Figure 1. Visual examples of our CDGNet in LIP database. Each class horizontal and vertical distribution labels that have been calculated from the ground truth of parsing. The generated labels can act as additional supervision signal to improve the parsing performance. Deeper color denotes higher probability of class existence.

Pseudo-code. PyTorch-like style pseudocode for generating class horizontal and vertical distribution. **—Algorithm 1**

```
# L, class_num:  Ground truth of parsing and the class number.
# G_{D_h}, G_{D_v}:  Class horizontal and vertical distribution labels.

# reshape the ground truth
h, w = L.shape
L=L.view(h*w)

# one-hot encoding of ground truth
M=torch.zero(h*w, class_num)
M.scatter_(1, L, 1)
M=M.transpose(0, 1)
M=M.view(class_num, h, w)

# Statistic on vertical direction
G_{D_h}=torch.zeros(class_number, w)
G_{D_h}=(torch.sum(M, dim=1)).float()
G_{D_h}[0]=0
max=torch.max(G_{D_h}, dim=1)[0]
max = max.unsqueeze(2)
G_{D_h}=G_{D_h}/(max+1e-5)

# Statistic on Horizontal direction
G_{D_v}=torch.zeros(class_number, h)
G_{D_v}=(torch.sum(M, dim=2)).float()
G_{D_v}[0]=0
max=torch.max(G_{D_v}, dim=1)[0]
max = max.unsqueeze(2)
G_{D_v}=G_{D_v}/(max+1e-5)

# return class horizontal and vertical distribution lables
return G_{D_h}, G_{D_v}
```

ing to the hats positions and coverage ranges of the two persons, which benefit parsing performance of this category. Although there are multiple persons in the same image, we can see from Fig. 2 that the class distribution traits successfully reflect each human part location and distribution range information. The effect of the distribution map is more significant for multiple human parsing than single human parsing. We achieve 4.5% higher than previous state-of-the-art performance in terms of mIoU, as viewed in Table 4 of the main body of the paper.

1.2. CDGNet Network Structure

The structure of CDGNet has been introduced in the main paper. We can divide the whole network into three parts: Horizontal and vertical feature learning, distribution signal supervision, and aggregating of distribution feature. The pseudocode is as Algorithm 2 shows. We argue that the network structure apply to any dataset in which each class has its solely distribution rule, not limited to human parsing datasets, such as face parsing, animal parsing, etc. We will confirm this in future.

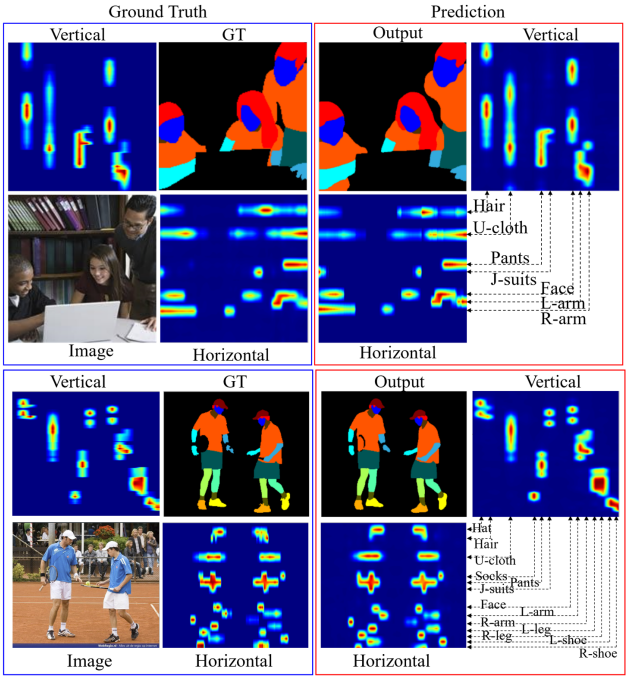

Figure 2. Visual examples of our CDGNet in CIHP database. Each class horizontal and vertical distributions in multiple human images. Deeper color denotes higher probability of class existence.

Horizontal and Vertical Feature Learning: To produce the required distribution feature, we leverage pooling operations on the input feature. There are two pooling techniques to select, average pooling and max pooling. To simulate the statistical operation, we adopt average pooling instead of max pooling. After the operation of pooling, we use a convolution with kernel size 3 activate by $relu$ function and decrease the channel number to half of the input channel. In our paper, the channel of input feature is 512, so, we decrease the number to 256.

Distribution Signal Supervision: We leverage the generated horizontal and vertical distribution of each category as supervision signal to make the network learning each class distribution prior. We utilize the 1D convolution of kernel size 3 and activate the feature by $sigmoid$ function. Here, the reason we adopt $sigmoid$ instead of $softmax$ is that there exits multiple classes in each row of horizontal and vertical distribution maps. It is possible that multiple

Pseudo-code. PyTorch-like style pseudocode for class distribution guided network. **—Algorithm 2**

```
# X_i: input feature
# X_o, P_h, P_v: class distribution guided feature, class horizontal and vertical distribution predictions.

#   Average pooling along horizontal and vertical directions
n,c,h,w=X_i.size()
Z_v=nn.AdaptiveAvgPool2d((1,w))(X_i).squeeze(2)
Z_h=nn.AdaptiveAvgPool2d((h,1))(X_i).squeeze(3)

# 1D convolution with kernel size 3 and relu activation apply to the feature. The channel number decreases from 512 to 256.
Z_h=conv_x3_relu(Z_h)
Z_v=conv_x3_relu(Z_v)

# achieve predictions of class horizontal and vertical distribution
P_h=conv_x3_sigmoid(Z_h)
P_v=conv_x3_sigmoid(Z_v)

# 1D convolution with kernel size 7 and sigmoid activation apply to the feature. The channel number increase from 256 to 512.
A_h=conv_x7_sigmoid(Z_h)
A_v=conv_x7_sigmoid(Z_v)

# upsampling the features P_h and P_v to have the same size with the input feature X_i
A'_h=upsampling(A_h,(h,w))
A'_v=upsampling(A_v,(h,w))
# aggregate horizontal and vertical distribution features, here α and β are two learnable parameters
A_d=α × A'_h + β × A'_v

# augment input feature using CDG feature and apply the convolution to the concatenated features
fea_aug = A_d × X_i
fea_cdg = torch.cat([X_i,fea_cdg, A_d],dim=1)
X_o=conv_3x3_relu(fea_cdg)

# return the output feature X_o, prediction of horizontal distribution P_h and vertical distribution P_v
return X_o, P_h, P_v
```

Table 1. Comparison for computational complexity at LIP dataset. * is the performance without the NLM module.

Method	Param	FLOPs	mIoU
Baseline (CE2P)	67.4M	80.8G	53.10
CorrPM (Human Pose)	71.5M	147.5G	55.33
Ours*	71.1M	81.7G	59.04
Ours	79.0M	130.7G	59.93

classes have high distribution values, simultaneously.

Aggregating of Distribution Feature: There are several strategies to aggregate different feature together, concatenation, addition, and multiplication. In our paper, we experimentally choose addition which is efficient and effective to fuse class horizontal and vertical distribution traits to achieve CDG feature. The channel number of CDG feature is 512 which is identical to the input feature.

Computational complexity: Table 1 shows the parameter comparison between the baseline, CorrPM and our network. Compared to the baseline, our full network (with NLM) requires +11.6M parameters and achieves $+6.83\%$ mIoU. While our method consumes almost the same parameters with CorrPM, we achieve $+3.7\%$ mIoU under much lower (-66.0G) FLOPs. When leveraging Non-Local module (NLM) that has been utilized in CorrPM in our method, we achieve significantly higher ($+4.6\%$) performance requiring little higher (7.9M) parameters, however, slightly lower (16.8G) FLOPs.

IoU loss and non-local module (NLM): We adopted the IoU loss and NLM for achieving the SOTA accuracy, because the others, e.g., SCHP and HHP, have used IoU loss or others such as CorrPM has leveraged NLM for improving their own accuracy. We make the ablation test, Table 2, on performances. When combined our method with IoU loss and NLM, we achieve a new state-of-the-art performance on LIP (60.30%), CIHP (65.56%) and ATR (97.39%) datasets in terms of mIoU and accuracy. As we analyzed in section 3.5 of the main paper, NLM consumes large memory footprints and has high computation complexity. If not leveraging NLM, our method achieves -0.89%, -0.05% and -0.47% performance in terms of mIoU and accuracy, respectively, which still is the highest performance compared to the previous methods, which verifies the superiority of our CDGNet.

Table 2. Effect of each component in LIP, ATR and CIHP. Single and Multi mean the single- and multi-scale tests, respectively.

DB	Metric	Multi	Single	Single without NLM
LIP	mIoU	60.30	59.93	59.04
ATR	Acc.	97.39	97.34	97.29
CIHP	mIoU	65.56	65.26	64.79

2. Our Novelty Compared to Human Pose-based Method

In essence, human pose and our class distribution all benefit human parsing. However, compared with previous works, e.g., using human pose, we have the following novelties; (1) Human pose provides the positions of *manually selected keypoints* while our method considers the distribution range and density of *whole categories, even hair, glasses, scarf*, in horizontal and vertical directions. (2) We make horizontal and vertical class distribution labels from the original GTs without any human labors, but in case of human pose or HHP, they require the human-annotated keypoints or hierarchical structures in advance. (3) Ours can be easily generalized for any additional classes that are not a human body and any tasks where pose information cannot be provided properly.

References

[1] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In *CVPR*, pages 770–778. IEEE, Jun. 2016. 1

[2] J. Lee, J. Yi, C. Shin, and S. Yoon. Bbam:bounding box attribution map for weakly supervised semantic and instance segmentation. In *CVPR*, pages 2643–2652, Jun. 2021. 1

[3] Z. Xingyi, W. Dequan, and P. Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019. 1

[4] B. Yan, X. Zhang, D. Wang, H. Lu, and X. Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *CVPR*, pages 5289–5298, Jun. 2021. 1