Partial Graph Reasoning for Neural Network Regularization Supplementary Materials

Tiange Xiang
University of Sydney
txia7609@uni.sydney.edu.au

Chaoyi Zhang
University of Sydney
chaoyi.zhang@sydney.edu.au

Yang Song
University of New South Wales
yang.song1@unsw.edu.au

Siqi Liu
Paige AI
lsqshr@gmail.com

Hongliang Yuan
Tencent AI Lab
haroldyuan@tencent.com

Weidong Cai
University of Sydney
tom.cai@sydney.edu.au

1 Experimental Details

1.1 Detailed Settings for Image Classification

CIFAR. We used the open-source PyTorch implementation [1] for all experiments on the CIFAR datasets. SGD with weight decay of 0.0005 and momentum of 0.9 was used as the optimizer. The batch size was set to 128 for both ResNet-50 and RegNetX-200MF backbones. The learning rate initially began at 0.1 and reduced by a factor of 0.1 at epochs 150 and 250 while all models were trained for 300 epochs from scratch. Standard augmentation techniques were adopted during training, including random cropping of a 32×32 sample from the 4-pixel padded images and random horizontal flipping with a probability of 0.5. All images were normalized by their mean and standard deviation in the pre-processing step.

ImageNet. For fair comparisons, we borrowed the implementations provided by [4] [2] and slightly change the total training epochs and optimizer scheduler to align with [1]. Specifically, we used SGD optimizer with weight decay of 0.0001 and momentum of 0.9 for optimizations. The batch size was set to 1024 for all experiments that were equally distributed on 8 V100 GPUs. All models were trained for 270 epochs starting with an initial learning rate of 0.1 and reduced by 0.1 at epoch 125, 200 and 250 without learning rate warming up. During training, input images are augmented by cropping into 224×224 patches and randomly horizontal flip. During validation, images are first scaled into 256×256 and then center cropped into 224×224 before being fed into the networks. The above training configurations are consistent to the ones used in [2, 1, 4].

1.2 Detailed Settings for Semantic Segmentation

Pascal VOC 2012. The experiments were conducted in a public framework [3]. We used the Adam optimizer [3] with 0.0001 weight decay to minimize the cross entropy loss. Learning rate starts from 0.001 and cosinanealing scheduled to $1e^{-5}$ in 400 epochs. The batch size was set to 64 for both backbones. Training images are randomly horizontal flipped, randomly scaled by a factor within [0.5, 1.5] and then center cropped leaving with 224×224 patches to fit the ResNet-50 backbone. Note that all backbone networks were trained from scratch rather than being pre-trained on ImageNet as suggested in [1].

MoNuSeg. Implementations from [5] [4] were adopted for the corresponding experiments. Cross entropy loss was minimized by the Adam optimizer with an initial learning rate of 0.01 and a decay rate of 0.00003 at per step. The training dataset was augmented through random rotation (with the angles from [-15, +15]), random x-y shifting (with the angles from [-5%, 5%]), random shearing,

[1] https://github.com/kuangliu/pytorch-cifar
[2] https://github.com/huawei-noah/Disout
[3] https://github.com/warmspringwinds/pytorch-segmentation-detection
[4] https://github.com/tiangexiang/BiO-Net

Preprint.

random zooming (within [0, 0.2]), and random flipping (both horizontally and vertically). The batch size was set to 2.

1.3 Detailed Settings for Point Cloud Analysis

We used the open-source implementation [5] for the point cloud classification experiments. We set the height and width of the 2D CNN feature maps of DropGraph and Disout to be equal to the number of points. SGD optimizer with weight decay of 0.0001 and momentum of 0.9 are used for the optimization. We utilized the cosineanealing scheduler to adjust the learning rate from 0.1 to 0.0001 in 250 epochs. The batch size was set to 32 for training and 16 for validation. Raw point clouds were first normalized into unit spheres followed by random scaling with a multiplier within [0.66, 1.5], random translation along the three directions by displacements within [-0.2, 0.2]. The number of neighbors in KNN was set to 20 for all experiments.

1.4 Detailed Settings for Graph Recognition

Cora. The official GCN implementation [6] was used for experiments on the Cora dataset. Adam optimizer with learning rate of 0.01 and weight decay of 0.0005 was used for optimization. For each run, we trained the models for 200 epochs without batch processing.

Protein. We used the open-source implementation [7] for experiments on the Protein dataset with the same training and validation split suggested in [6]. The optimizer and network were set identically to the ones used for Cora experiments, except that we place an additional linear layer at the end of the network. Learning rate was set to 0.02 for training on the entire dataset for 200 epochs.

2 ImageNet Training Curves

Here we compare the accuracy and loss curves between DropGraph and Disout during ImageNet training in Figure 1. Validation and training statistics are plotted in solid lines and dotted lines respectively, with red denote our DropGraph and blue Disout.

An effective regularizer is able to alleviate over-fitting on the training set and improve the generalization ability on the validation set. According to Figure 1 left, DropGraph yields generally lower training accuracy and higher validation accuracy. As shown in Figure 1 right, DropGraph yields higher training loss with lower validation loss. Therefore, our DropGraph demonstrates stronger regularization effects than Disout.

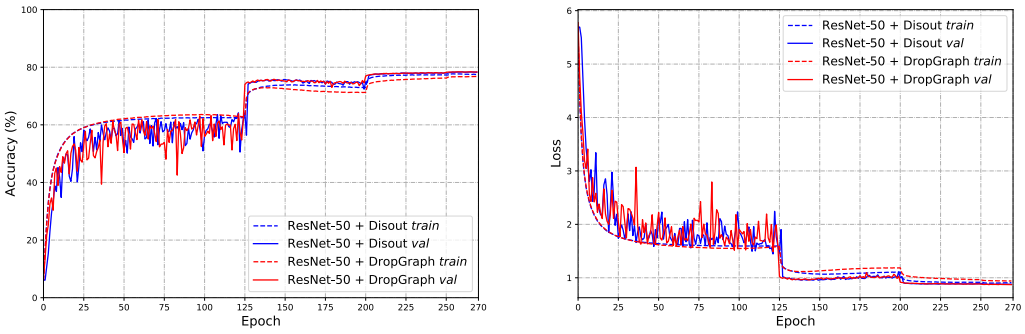

Figure 1: **Left:** Accuracy curves. **Right:** Loss curves.

[5] https://github.com/WangYueFt/dgcnn
[6] https://github.com/tkipf/pygcn
[7] https://github.com/cszhangzhen/HGP-SL

3 Deployment Positions of DropGraph

In Figure 2, we show where to apply DropGraph on ResNet and U-Net style networks. As mentioned in main paper Sec. 3.2. and Sec. 4., in general, we apply DropGraph after each activation layer. Note that we also apply distortions to skip features in all networks with residual connections.

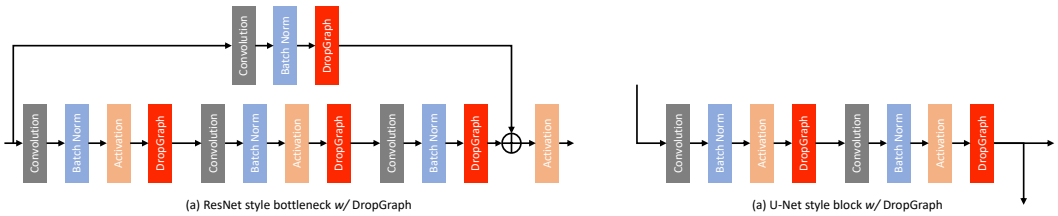

Figure 2: **Deployment position of DropGraph on ResNet and U-Net.**

4 More Semantic Segmentation Results

We provide more qualitative semantic segmentation results in Figure 3. Pascal VOC results are obtained from the FCN backbone and MoNuSeg results are collected from the U-Net backbone. Clearly, the backbone networks generate the most accurate segmentation masks by applying our proposed DropGraph.

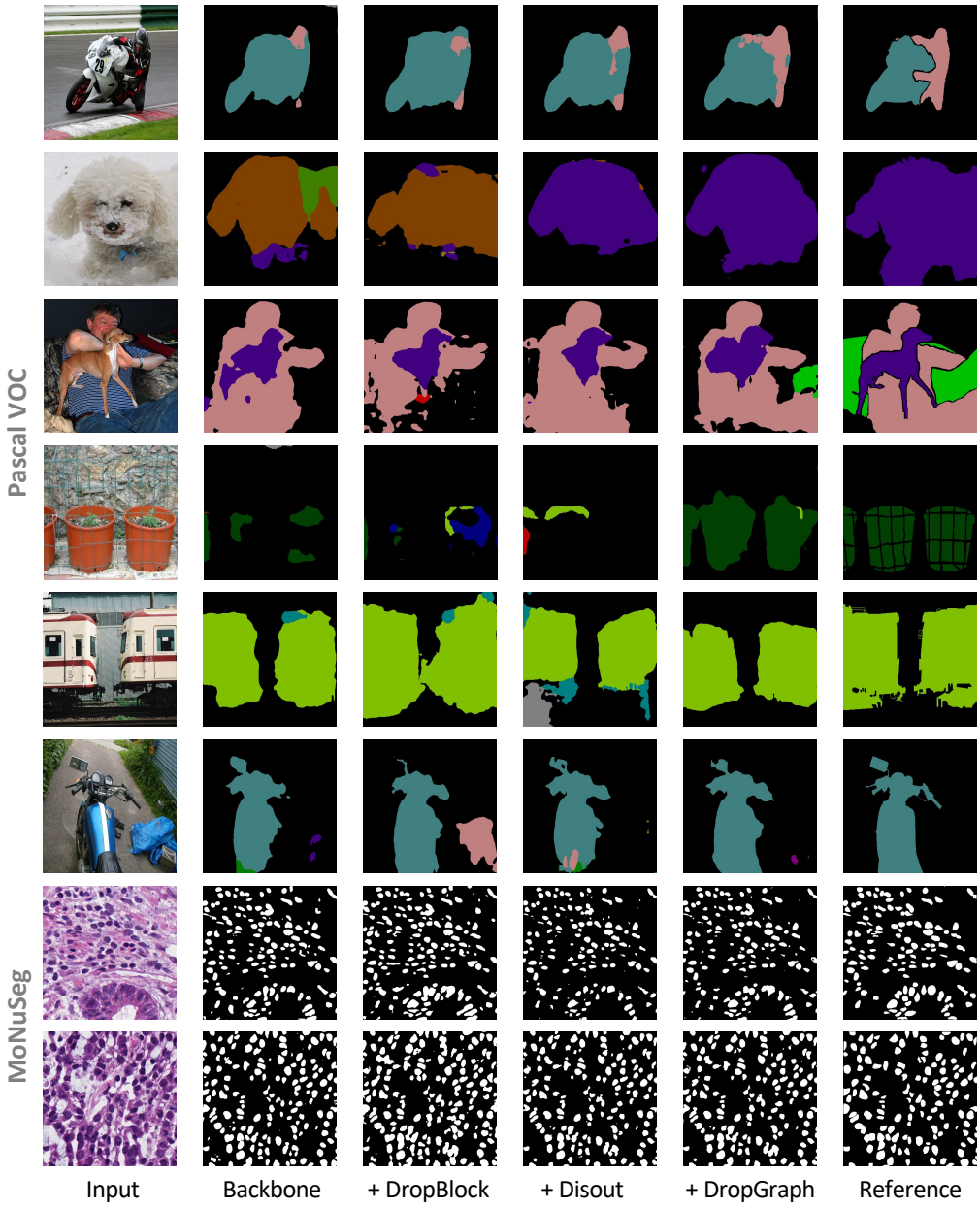

Figure 3: **More qualitative results on semantic segmentation.**

References

[1] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10727–10737, 2018.

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

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] Yehui Tang, Yunhe Wang, Yixing Xu, Boxin Shi, Chao Xu, Chunjing Xu, and Chang Xu. Beyond dropout: Feature map distortion to regularize deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[5] Tiange Xiang, Chaoyi Zhang, Dongnan Liu, Yang Song, Heng Huang, and Weidong Cai. Bio-net: Learning recurrent bi-directional connections for encoder-decoder architecture. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 74–84. Springer, 2020.

[6] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019.