

000

Supplementary for Dual Adaptive 001 Transformations for Weakly Supervised Point 002 Cloud Segmentation

003

004

005 Zhonghua Wu^{1,2}, Yicheng Wu³, Guosheng Lin^{*1,2}, Jianfei Cai^{2,3}, and Chen
006 Qian⁴

007

008 ¹ S-lab, Nanyang Technological University

009 ² School of Computer Science and Engineering, Nanyang Technological University

010 ³ Department of Data Science and AI, Monash University

011 ⁴ SenseTime Research

012 zhonghua001@e.ntu.edu.sg

013

014

A More Ablation Studies

015

016 **Different hyper-parameters α and β .** We conduct experiments with different
017 α and β under the OTOC setting on the S3DIS dataset. From Table A, we can
018 see that setting both α and β as 2 yields the best performance for the weakly
019 supervised point cloud segmentation task.

020

021 **Table A.** Ablation studies on different α and β under the OTOC setting on the S3DIS
022 dataset.

023

α	β	mIoU(%)
	1	55.6
2	2	56.5
	5	56.2

α	β	mIoU(%)
1		55.1
2	2	56.5
5		56.1

031 **Different Superpoint in RAD.** We conduct experiments to analyze the sensi-
032 tivity of superpoints, which are used in RAD. Specifically, we generate the
033 superpoints with different hyperparameters of the number of neighbors for the
034 geometric features *geo* and adjacency graph *adj*. As shown in Table B, we can
035 see that our RAD module is not very sensitive to different superpoints.

036

037

B Psudeocodes

038

039 Algorithm A and B respectively provide the pseudocodes of our designed LAP
040 and RAD modules.

041

042 * Corresponding author: G. Lin (e-mail: gslin@ntu.edu.sg)

043

044 000
001 001
002 002
003 003
004 004
005 005
006 006
007 007
008 008
009 009
010 010
011 011
012 012
013 013
014 014
015 015
016 016
017 017
018 018
019 019
020 020
021 021
022 022
023 023
024 024
025 025
026 026
027 027
028 028
029 029
030 030
031 031
032 032
033 033
034 034
035 035
036 036
037 037
038 038
039 039
040 040
041 041
042 042
043 043
044 044

Table B. Ablation studies on different $geof$ and adj under the OTOC setting on the S3DIS dataset.

$geof$	adj	mIoU(%)
30	10	54.4
45	5	54.7
45	10	56.5

C More Qualitative Results

Figure A and B respectively show five more segmentation results obtained by our proposed model on the S3DIS and ScanNet-v2 dataset. Our DAT model is able to generate accurate segmentation masks for most of the points only with the weak annotation training.

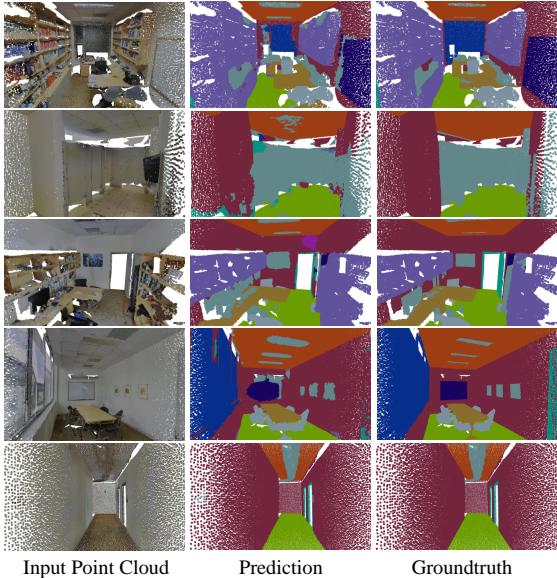


Fig. A. Visualization results obtained by our DAT model on the S3DIS dataset under the “OTOC” setting.

090
091
092
093
094
095
096
097

098 **Algorithm A** Pseudocode of Local Adaptive Perturbation (LAP) module in a
099 PyTorch-like style.

100 # CPG: class-aware perturbation generator
101 # c: point coordinates
102 # f: point features
103 # xi_c: theta for coordinates
104 # xi_f: theta for features
105 # eps_c: epsilon for coordinates
106 # eps_f: epsilon for features
107 # ip: iteration times of computing adv noise (default: 1)

108 import torch.nn.functional as F
109
110 def LAP(c, f, model):
111
112 pred = model(c, f)
113
114 pseudo_label = pred.max(dim=1)[1] # generate pseudo label
115 CPG.update_CV(c, f, pseudo_label) # update the covariance matrices
116
117 c_init, f_init = CPG.generator(c, f, pseudo_label) # generate initial unit vectors for
118 coordinates and features
119
120 # normalize the initial unit vectors
121 c_init = _l2_normalize(c_init)
122 f_init = _l2_normalize(f_init)
123
124 for _ in range(ip):
125 c_init.require_grad()
126 f_init.require_grad()
127
128 pred_init = model(c + xi_c * c_init, f + xi_f * f_init)
129 adv_distance = F.kl_div(F.log_softmax(pred_init, dim=1), pred)
130
131 # generate adversarial perturbations
132 adv_distance.backward()
133
134 c_adv = _l2_normalize(c_init.grad)
135 f_adv = _l2_normalize(f_init.grad)
136
137 model.zero_grad()
138
139 pred_hat = model(c + eps_c * c_adv, f + eps_f * f_adv)
140 Loss_pc = F.kl_div(F.log_softmax(pred_hat, dim=1), pred) # point-level consistency loss

127
128
129
130
131
132
133
134

135
136
137
138
139
140
141
142
143
144
145 **Algorithm B** Pseudocode of Regional Adaptive Deformation (RAD) module
146 in a PyTorch-like style.
147

```
148     # c: point coordinates
149     # f: point features
150     # xi: theta
151     # eps: epsilon
152     # ip: iteration times of computing adv noise (default: 1)
153
154     import torch.nn.functional as F
155
156     def RAD(c, f, model):
157
158         pred = model(c, f)
159
160         S = SP_G(c, f) # offline superpoint extraction
161
162         A_init = A_generator(c, S) # generate initial affine transformation matrices
163         A_init = normalize(A_init) # normalize the matrices
164
165         for _ in range(ip):
166             A_init.require_grad_()
167
168             c_init = affine(c, S, A_init) # generate initial perturbed point cloud
169
170             pred_init = model(c_init, f)
171
172             adv_distance = F.kl_div(F.log_softmax(pred_init, dim=1), pred)
173
174             # generate region-level adversarial examples
175             adv_distance.backward()
176             A_adv = normalize(A_init.grad)
177             model.zero_grad()
178
179             c_adv = affine(c, S, eps * A_adv)
180             pred_hat = model(c_adv, f)
181
182             Loss_rc = F.kl_div(F.log_softmax(pred_hat, dim=1), pred) # region-level consistency loss
```

170
171
172
173
174
175
176
177
178
179

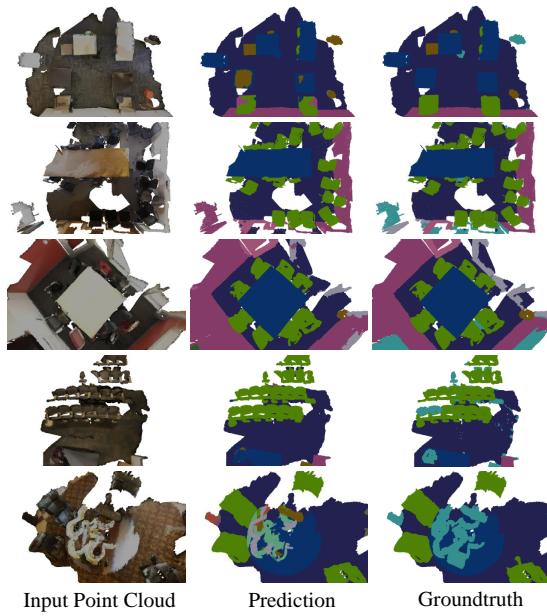


Fig. B. Visualization results obtained by our DAT model on the ScanNet-v2 dataset under the “20 points” setting.