Computer Vision Reading Group

PointDAN: A Multi-Scale 3D Domain Adaption Network for Point Cloud Representation - Qin, Can, et al.

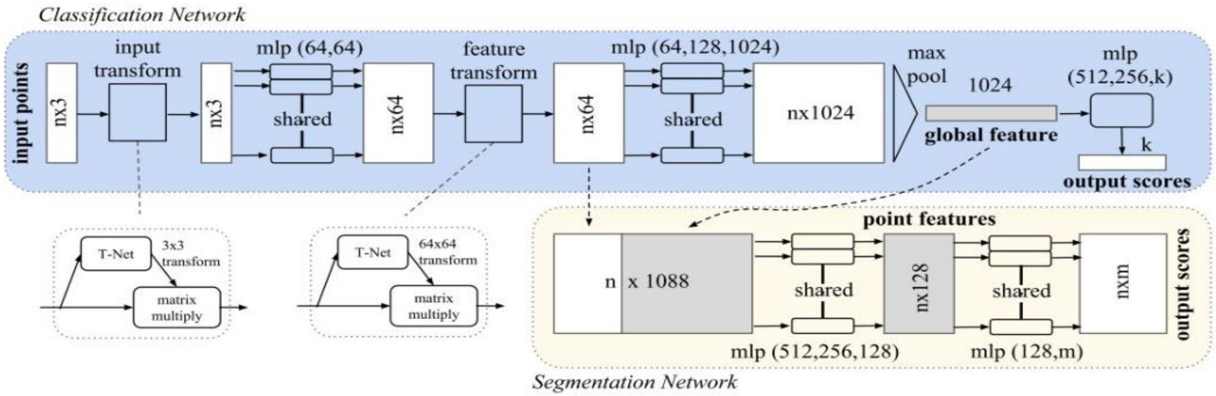
April 21st 2020 – Xiaomeng (Mandy) Zhu

Background

- 3D vision
  - multi-view, voxel, grid, 3D mesh and point cloud
- Point Cloud
  - straightforward representation
  - Properties:
    - > *Unordered*
    - > *Interaction among points*
    - > *Invariance under transformations*

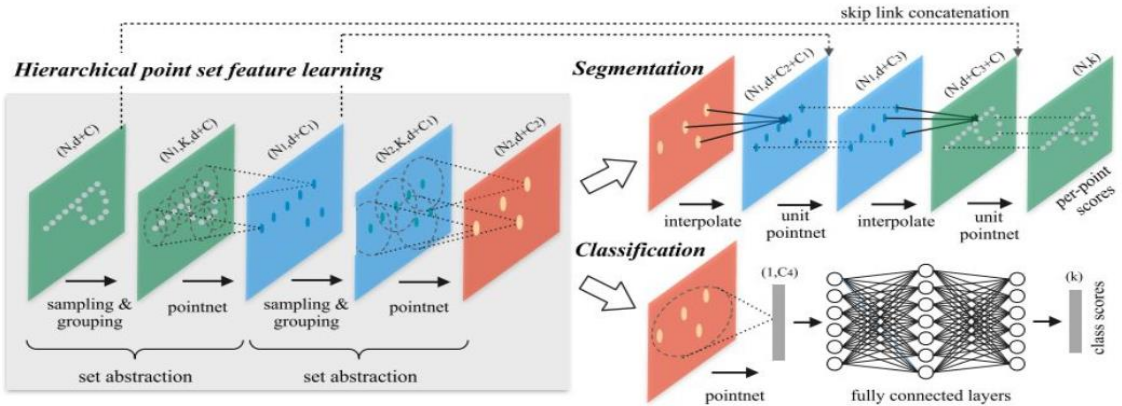
Background

- PointNet (https://arxiv.org/abs/1612.00593)
  - First deep neural networks directly deal with point clouds
  - Proposes a symmetry function and a spatial transform network to obtain the invariance to point permutation.
  - Local geometric information is ignored



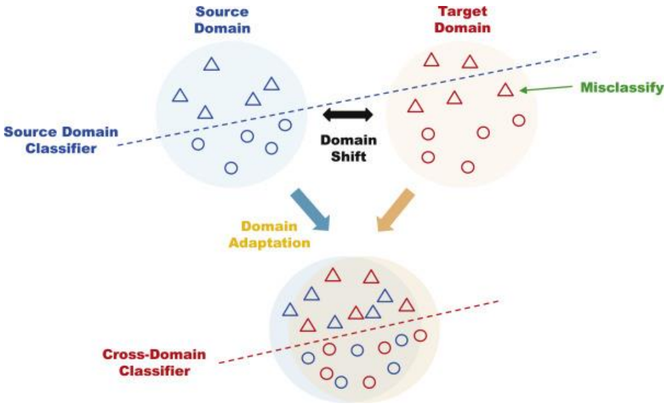
Background

- PointNet++ (https://arxiv.org/abs/1706.02413)
  - Focus on how to effectively utilize local feature.
  - Sampling (farthest point sampling, FPS)
  - Grouping
  - Feature Learning



Background

- Unsupervised Domain Adaptation
  - Narrow the distribution shift between the target and source domain
  - Match either the marginal distribution or the conditional distribution between domains via feature alignment
  - Learning a mapping function f which projects the raw image features into a shared feature space across domains.
  - Maximizing the inter-class discrepancy while minimize the intra-class distance in a subspace simultaneously.



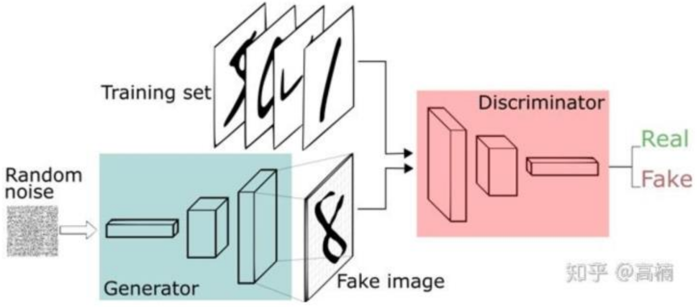
Background

- MMD (https://www.ncbi.nlm.nih.gov/pubmed/16873512)
  - maximum mean discrepancy
  - k: Gaussain kernel function RBF

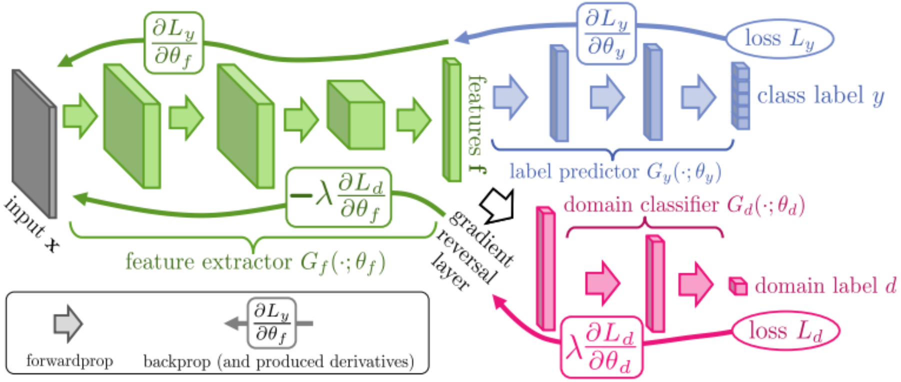
$$MMD^2[F,p,q] = \frac{1}{m(m-1)}\sum_{i\neq j}^{m} k(x_i,\ x_j) + \frac{1}{n(n-1)}\sum_{i\neq j}^{n} k(y_i,\ y_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} k(x_i,\ y_j)$$

Background

- Domain-Adversarial Training of Neural Networks DANN (https://arxiv.org/abs/1505.07818)
  - Generator becomes a feature extractor
  - fixed feature representations becomes transferable features
  - Domain – invariance
  - Discriminativeness



GAN

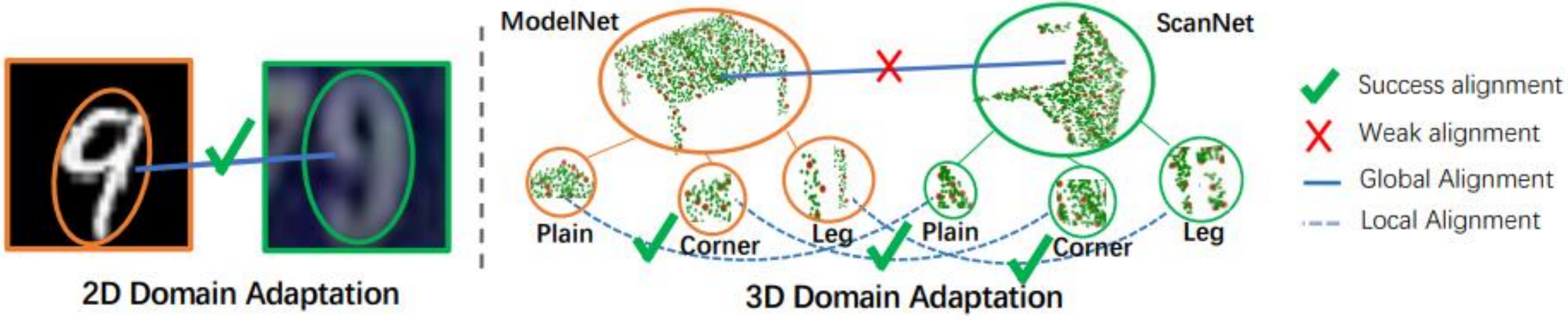


DANN

PointDAN

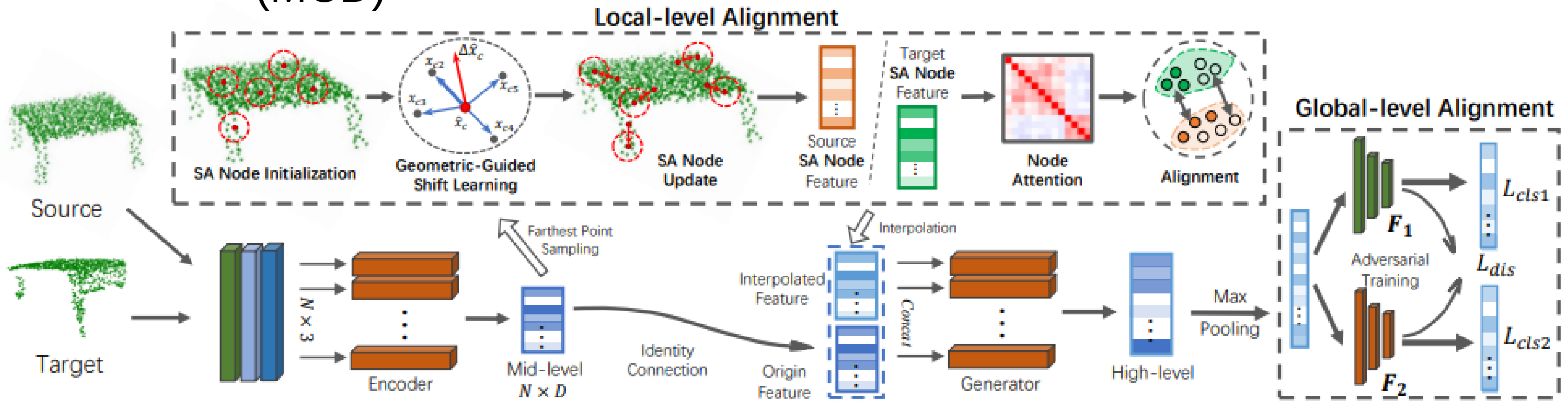
3D point cloud domain adaptation

- Abundant spatial geometric information
- Local Alignment
- Global Alignment

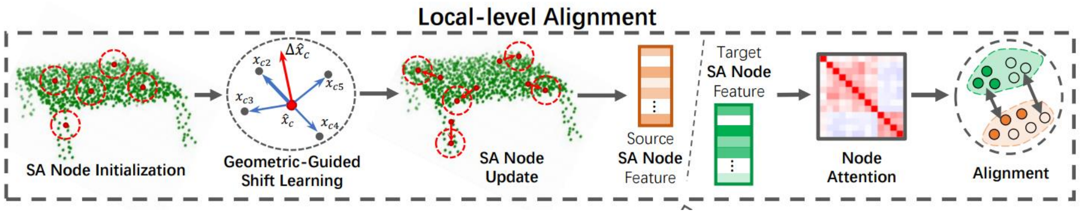


Method

- Local Alignment
  - Self-Adaptive (SA) node module with an adjusted receptive field
- Global Alignment
  - An Adversarial-training strategy: Maximum Classifier Discrepancy (MCD)



Local Alignment



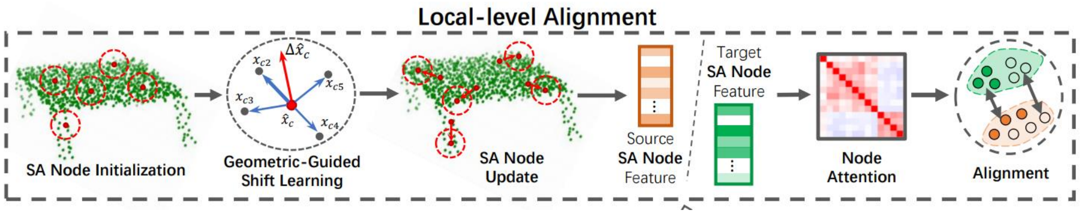
Local Alignment:

1. Initialize the location of node by farthest point sampling to get n nodes and their k nearest neighbor points

   {Sc|Sc = {xˆc, xc1, ..., xck}, x ⊆ R 3} n c=1
   where the c-th region Sc contains a node xˆc and its surrounding k nearest neighbor points {xc1, ..., xck}.

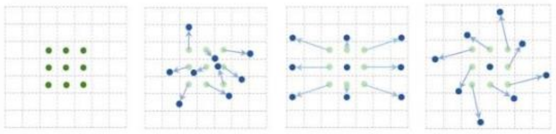
2. Apply the bottom 3 feature extraction layers of PointNet as the encoder E, extracted the mid-level point feature from the encoder v = E(x|ΘE) to get vˆc and {vc1, …, vck}

Local Alignment

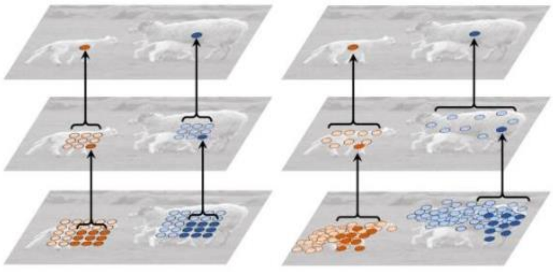


Local Alignment:
3. Geometric – Guided Shift Learning

- Deformable convolution network



(a) (b) (c) (d)



(a) standard convolution (b) deformable convolution

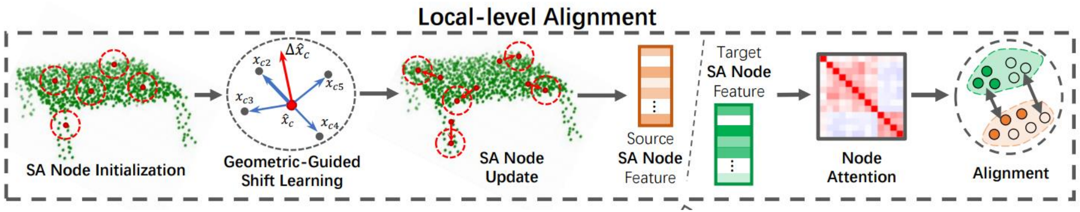
- Utilize the local edge vector as a guidance during learning

offset calculate:

$$\Delta\hat{x}_c = \frac{1}{k}\sum_{j=1}^{k}\left(R_T(\mathbf{v}_{cj} - \hat{\mathbf{v}}_c)\cdot\underbrace{(x_{cj} - \hat{x}_c)}_{\text{the edge direction}}\right),$$

RT is the weight from one convolution layer for transforming feature

Local Alignment



4. Self-adaptive update of node and find their new k nearest neighbor points:

$$\hat{x}_c = \hat{x}_c + \Delta\hat{x}_c,$$

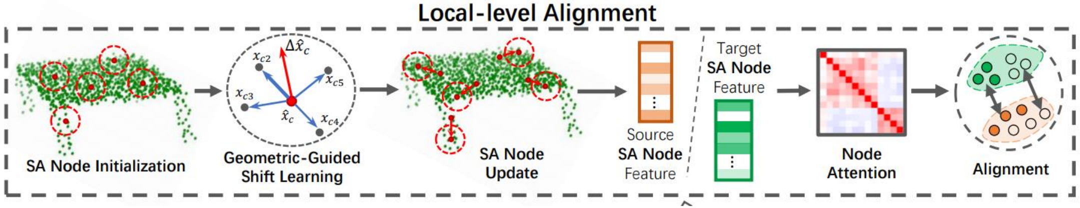
$$\{x_{c1}, ..., x_{ck}\} = kNN(\hat{x}_c | x_j, j = 0, ..., M - 1).$$

5. Compute the final node features vˆc by gathering all the point features inside their regions:

$$\hat{\mathbf{v}}_c = \max_{j=1,..,k} R_G(\mathbf{v}_{cj}).$$

where RG is the weight of one convolution layer for gathering point features in which RG S RT = R.

Local Alignment



6. Apply node attention network with residual structure to model the contribution of each SA node for alignment

- Channel Attention (CA) from RCAN Image Super-Resolution Using Very Deep Residual Channel Attention Networks (https://arxiv.org/abs/1807.02758)

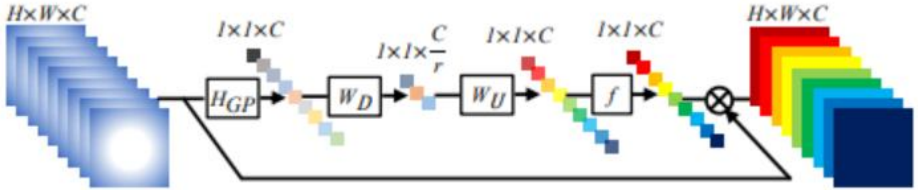
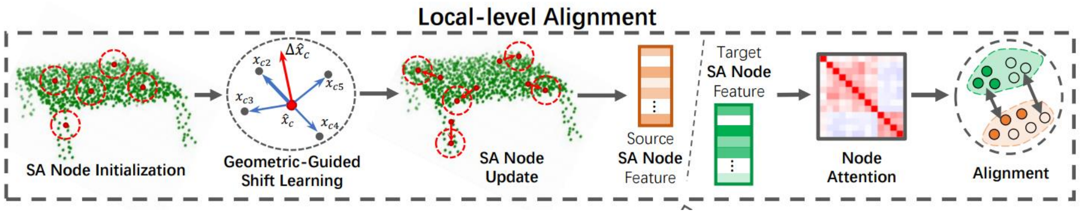


Fig. 3. Channel attention (CA). ⊗ denotes element-wise product

$$\mathbf{h}_c = \varphi(W_U \delta(W_D \mathbf{z}_c)) \cdot \hat{\mathbf{v}}_c + \hat{\mathbf{v}}_c,$$

where zc = E(vˆc(k)) indicates the mean of the c-th node feature. δ(·) and ϕ(·) represent the ReLU function and Sigmoid function respectively. WD is the weight set of a convolutional layer with 1 × 1 kernels, which reduces the number of channels with the ratio r. The channel-upscaling layer WU , where WU S WD = W, increases the channels to its original number with the ratio r.

Local Alignment



7. Local alignment by minimize the MMD loss

$$L_{mmd} = \frac{1}{n_s n_s} \sum_{i,j=1}^{n_s} \kappa(\mathbf{h}_i^s, \mathbf{h}_j^s) + \frac{1}{n_s n_t} \sum_{i,j=1}^{n_s, n_t} \kappa(\mathbf{h}_i^s, \mathbf{h}_j^t) + \frac{1}{n_t n_t} \sum_{i,j=1}^{n_t} \kappa(\mathbf{h}_i^t, \mathbf{h}_j^t),$$

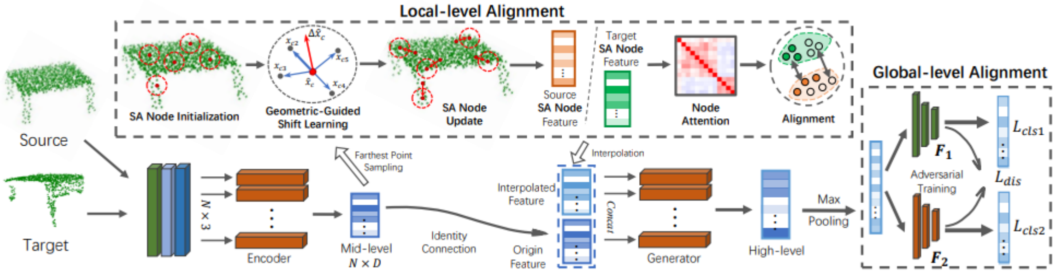
where κ is a kernel function:Radial Basis Function (RBF)

$$MMD^2[F,p,q] = \frac{1}{m(m-1)} \sum_{i \neq j}^{m} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i \neq j}^{n} k(y_i, y_j) - \frac{2}{mn} \sum_{i,j=1}^{m,n} k(x_i, y_j)$$

Joint Distribution adaptation: to achieve effective and robust transfer learning, aim to simultaneously minimize the differences in both the marginal distributions and conditional distributions across domains.

$$\min_{\mathbf{A}^{\mathrm{T}}\mathbf{X}\mathbf{H}\mathbf{X}^{\mathrm{T}}\mathbf{A}=\mathbf{I}} \sum_{c=0}^{C} \operatorname{tr}\left(\mathbf{A}^{\mathrm{T}}\mathbf{X}\mathbf{M}_c\mathbf{X}^{\mathrm{T}}\mathbf{A}\right) + \lambda \|\mathbf{A}\|_F^2 \quad (M_c)_{ij} = \begin{cases} \frac{1}{n_s^{(c)} n_s^{(c)}}, & \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_s^{(c)} \\ \frac{1}{n_t^{(c)} n_t^{(c)}}, & \mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_t^{(c)} \\ \frac{-1}{n_s^{(c)} n_t^{(c)}}, & \begin{cases} \mathbf{x}_i \in \mathcal{D}_s^{(c)}, \mathbf{x}_j \in \mathcal{D}_t^{(c)} \\ \mathbf{x}_j \in \mathcal{D}_s^{(c)}, \mathbf{x}_i \in \mathcal{D}_t^{(c)} \end{cases} \\ 0, & \text{otherwise} \end{cases}$$

Global Alignment



Global Alignment:

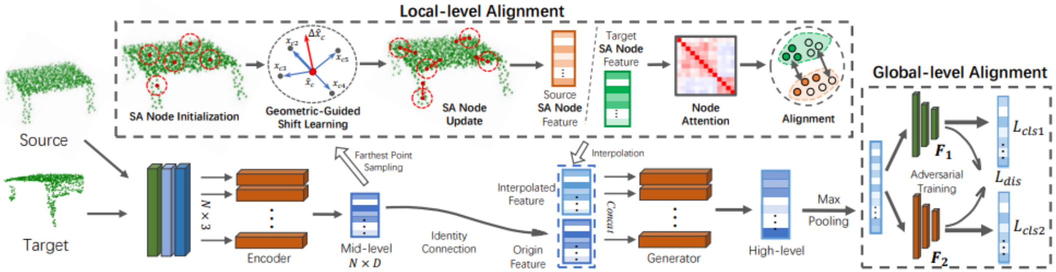
8. Apply Generator (feature extractor):

- Apply the same encoder E to extract raw point cloud features: h˜ i = E (xi |ΘE) over the whole object.
- Concatenated the point features with interpolated SA-node features as hˆi = [hi , h˜ i] to capture the geometry information in multi-scale.
- Use the final convolution layer (i.e., conv4) of PointNet as the generator network G, feed the hˆI to G, then apply max-pooling, to make the feature to a high-level global feature

$$\mathbf{f}_i = max - pooling(G(\hat{\mathbf{h}}_i|\Theta_G)),$$

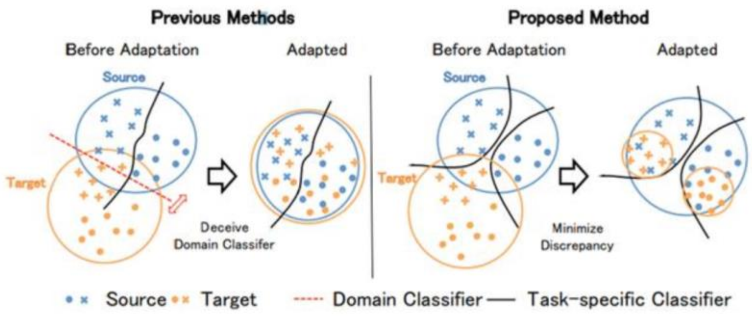
where fi ∈ R d represents the global feature of the i-th sample. And d is usually assigned as 1,024

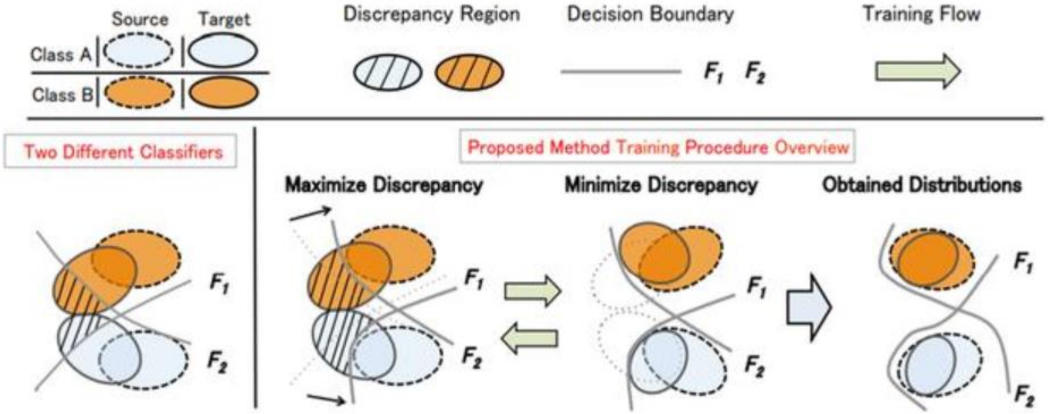
Global Alignment



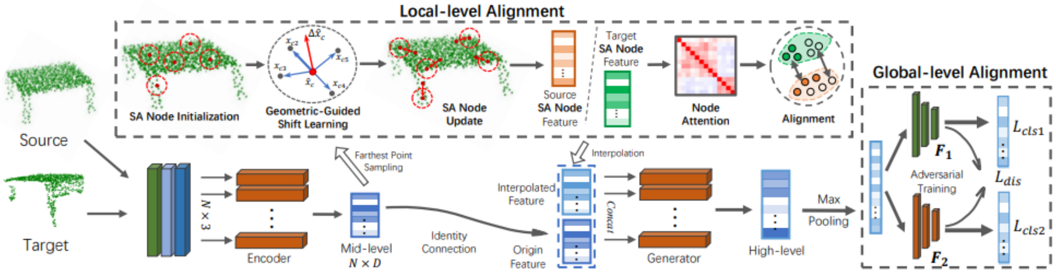
Maximum Classifier Discrepancy MCD (https://arxiv.org/abs/1712.02560)

- Utilizing the task-specific decision boundaries between classes.
- Match the feature distributions between different domains
- Two classifier networks F1 and F2 as discriminator.





Global Alignment



Maximum Classifier Discrepancy MCD (https://arxiv.org/abs/1712.02560)

- Training on MCD

method

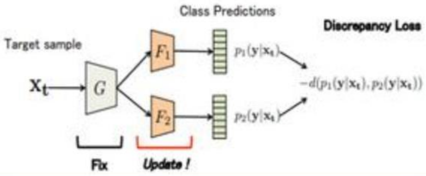
Step A: train classifier and generator

minimize softmax cross entropy

$$\min_{G,F_1,F_2} \mathcal{L}(X_s, Y_s).$$

$$\min \mathcal{L}(X_s, Y_s) = -\mathbb{E}_{(\mathbf{x}_s, y_s)\sim(X_s,Y_s)} \sum_{k=1}^{K} \mathbb{I}_{[k=y_s]} \log p(\mathbf{y}|\mathbf{x}_s)$$

Step B : Maximize discrepancy on target (Fix G)

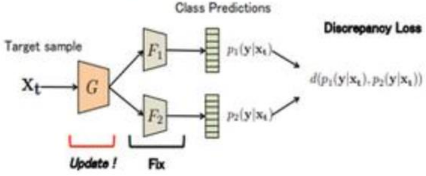


$$\min_{F_1,F_2} \mathcal{L}(X_s, Y_s) - \mathcal{L}_{\text{adv}}(X_t).$$

$$\mathcal{L}_{\text{adv}}(X_t) = \mathbb{E}_{\mathbf{x_t}\sim X_t}[d(p_1(\mathbf{y}|\mathbf{x_t}), p_2(\mathbf{y}|\mathbf{x_t}))]$$

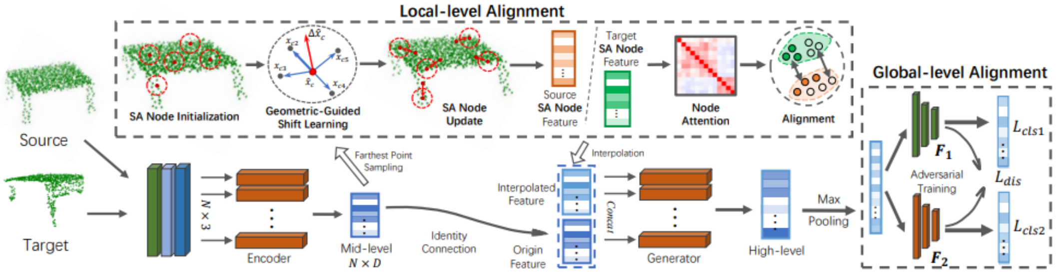
$$d(p_1, p_2) = \frac{1}{K}\sum_{k=1}^{K} |p_{1k} - p_{2k}|,$$

Step C : Minimize discrepancy on target (Fix F_1, F_2)



$$\min_{G} \mathcal{L}_{\text{adv}}(X_t).$$

Global Alignment



9. Apply Maximum Classifier Discrepancy MCD

- Task loss: classify loss:

$$L_{cls}(X_s, Y_s) = -\mathbb{E}_{(\mathbf{x}_s, y_s)\sim(X_s, Y_s)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log(p((\mathbf{y} = y_s)|\underbrace{G(\underbrace{E(\mathbf{x}_s|\Theta_E)}_{\text{encoder extract feature hs}}|\Theta_G)}_{\text{generator to high-level feature fs}})).$$

- Discrepancy loss: l1 distance between the SoftMax scores of two classifier:

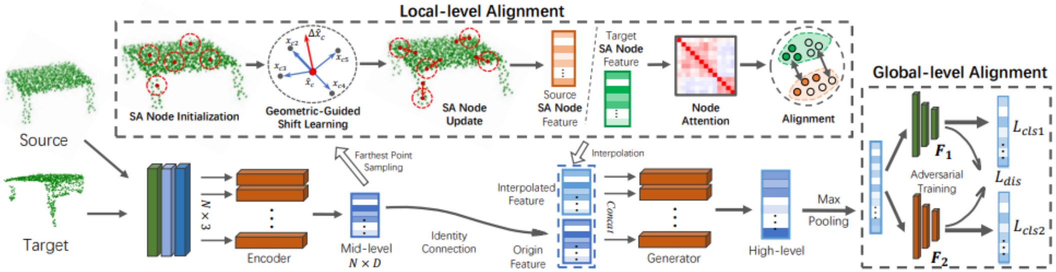
$$L_{dis}(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_t \sim X_t}[|p_1(\mathbf{y}|\mathbf{x}_t) - p_2(\mathbf{y}|\mathbf{x}_t)|].$$

The two classifiers F1 and F2 take the features fi and classify them into K classes as

$$p_j(\mathbf{y_i}|\mathbf{x}_i) = F_j\left(\mathbf{f}_i|\Theta_F^j\right)$$

where j = 1, 2, pj (yi |xi) is the K-dimensional probabilistic softmax results of classifiers

Global Alignment



10. Training

- Step 1, minimize the classification loss to minimize empirical risk on source domain, maximize discrepancy loss to train F1, F2

$$\min_{F_1, F_2} L_{cls} - \lambda L_{dis}.$$

- Step 2, minimize discrepancy loss, classification loss, and MMD loss to train generator G, encoder E, node attention network W and transform network R.

$$\min_{G, E, \mathcal{W}, \mathcal{R}} L_{cls} + \lambda L_{dis} + \beta L_{mmd},$$

where both λ and β are hyper-parameters which manually assigned as 1.

Theoretical analysis

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F. and Vaughan, J.W., 2010. A theory of learning from different domains. *Machine learning*, *79*(1-2), pp.151-175.

- MCD method is motivated by this theory.
- use H-divergence to establish a connection between source domain error and target domain error:

$$\epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) + C.$$

$$\Downarrow$$

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) = 2 \sup_{h_1, h_2 \in \mathcal{H}} \left| \mathbb{E}_{\mathbf{x} \sim \mathcal{S}} \mathbb{1}_{[h_1(\mathbf{x}) \neq h_2(\mathbf{x})]} - \mathbb{E}_{\mathbf{x} \sim \mathcal{T}} \mathbb{1}_{[h_1(\mathbf{x}) \neq h_2(\mathbf{x})]} \right|.$$

$$\Downarrow$$

$$\sup_{h_1, h_2 \in \mathcal{H}} \mathbb{E}_{\mathbf{x} \sim \mathcal{T}} \mathbb{1}_{[h_1(\mathbf{x}) \neq h_2(\mathbf{x})]} = \sup_{F_1, F_2} \mathbb{E}_{\mathbf{x} \sim \mathcal{T}} \mathbb{1}_{[F_1 \circ G(\mathbf{x}) \neq F_2 \circ G(\mathbf{x})]}.$$

$$\Downarrow$$

$$\min_{G} \max_{F_1, F_2} \mathbb{E}_{\mathbf{x} \sim \mathcal{T}} \mathbb{1}_{[F_1 \circ G(\mathbf{x}) \neq F_2 \circ G(\mathbf{x})]}.$$

PointDA – 10 Dataset

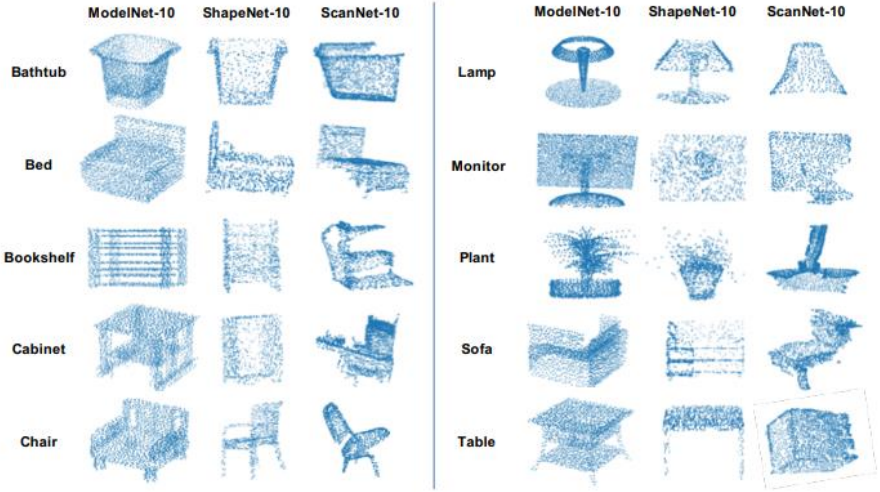


Table 1: Number of samples in proposed datasets.

Dataset		Bathtub	Bed	Bookshelf	Cabinet	Chair	Lamp	Monitor	Plant	Sofa	Table	Total
M	Train	106	515	572	200	889	124	465	240	680	392	4, 183
	Test	50	100	100	86	100	20	100	100	100	100	856
S	Train	599	167	310	1, 076	4, 612	1, 620	762	158	2, 198	5, 876	17, 378
	Test	85	23	50	126	662	232	112	30	330	842	2, 492
S*	Train	98	329	464	650	2, 578	161	210	88	495	1, 037	6, 110
	Test	26	85	146	149	801	41	61	25	134	301	1, 769

- ModelNet-10 (M): Sample points on the surface as pointNet++ to fully cover the CAD models;
- ShapeNet-10(S): Apply uniform sampling to collect the points of ShapeNet on surface, which may lose some marginal points compare to ModelNet;
- ScanNet-10(S*): Isolate from real-world indoor scenes, the objects often loss some parts and get occluded by surroundings.

Experiments

- Six types of adaptation scenarios which are M → S, M → S*, S → M, S → S*, S* → M and S* → S
- PointNet as backbone of Encoder E and Generator G;
- F1 and F2 are two-layer multilayer perceptron (MLP);
- Optimizer: PyTorch with Adam (A method for stochastic optimization);
- GPU: NVIDIA TITAN GPU;
- Learning rate: 0.0001 under weight decay 0.0005;
- Epochs = 200;
- Batch size = 64;
- Extract the SA node features from conv3, number of SA node = 64.

Results

Table 2: Quantitative classification results (%) on PointDA-10 Dataset.

	G	L	A	P	M→S	M→S*	S→M	S→S*	S*→M	S*→S	Avg
w/o Adapt					42.5	22.3	39.9	23.5	34.2	46.9	34.9
MMD [18]	√				57.5	27.9	40.7	26.7	47.3	54.8	42.5
DANN [10]	√				58.7	29.4	42.3	30.5	48.1	56.7	44.2
ADDA [32]	√				61.0	30.5	40.4	29.3	48.9	51.1	43.5
MCD [26]	√				62.0	31.0	41.4	31.3	46.8	59.3	45.3
Ours	√	√			62.5	31.2	41.5	31.5	46.9	59.3	45.5
	√	√	√		63.7	32.1	44.5	33.7	48.2	63.0	47.5
	√	√	√	√	**64.2**	**33.0**	**47.6**	**33.9**	**49.1**	**64.1**	**48.7**
Supervised					90.5	53.2	86.2	53.2	86.2	90.5	76.6

- Ablation Study: global feature alignment, i.e., G, local feature alignment, i.e., L, SA node module (including adaptive offset and attention), i.e., A, and the self-training, i.e., P: to finetune the model with 10% pseudo target labels generated from the target samples with the highest SoftMax scores.
- Outperform (especially on A)
- Great margin exist between supervised method and DA methods
- MMD and GAN-based methods

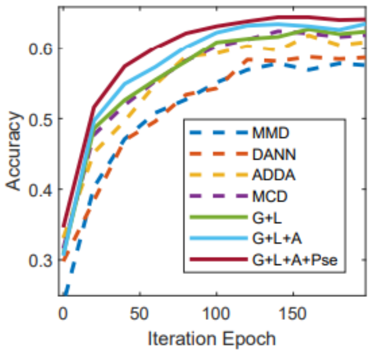
Results

Table 3: Class-wise classification results (%) on ModelNet to ShapeNet.

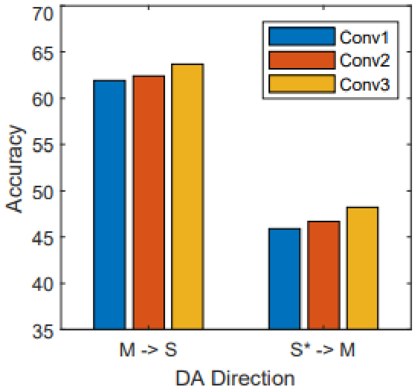
	G	L	A	P	Bathtub	Bed	Bookshelf	Cabinet	Chair	Lamp	Monitor	Plant	Sofa	Table	Avg
w/o Adapt					59.4	1.0	18.4	7.4	55.7	43.5	84.8	60.0	3.4	39.7	37.3
MMD [18]	√				77.1	0.7	20.0	1.6	63.6	58.4	88.8	83.4	0.5	**87.6**	48.2
DANN [10]	√				82.6	0.4	20.1	1.5	72.1	52.6	90.2	86.7	1.0	80.2	48.6
ADDA [32]	√				84.5	1.0	**22.9**	2.4	66.7	62.8	83.6	70.1	1.8	86.8	48.3
MCD [26]	√				84.8	**4.4**	18.4	**7.7**	74.9	62.0	85.6	80.0	1.6	82.2	50.2
Ours	√	√			84.6	0.8	19.2	1.6	75.6	61.2	**92.7**	**86.3**	0.9	83.4	50.6
	√	√	√		**85.7**	2.4	20.4	1.0	79.0	**64.2**	90.1	83.3	**3.6**	83.0	**51.3**
	√	√	√	√	84.7	1.6	19.0	1.3	**81.9**	63.3	90.5	82.3	2.2	82.9	51.0
Supervised					88.9	88.6	47.8	88.0	96.6	90.9	93.7	57.1	92.7	91.1	83.5

- Local alignment help boost the performance on most of the class
- Imbalanced training sample affect the performance of models, and self-training

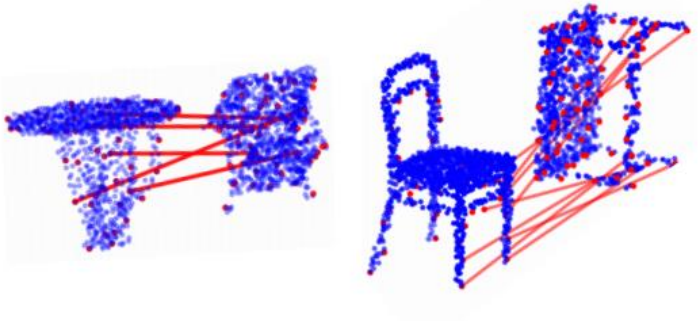
Results



Convergence M → S:
local alignment helps accelerate the convergence and make them more stable



Mid-level feature extraction from different conv layer:
conv3 contains the best mid-level feature for local alignment



Local alignment of two cross-domain objects:
SA nodes represent similar geometry structure, i.e., legs, plains contribute most to local alignment.
Common knowledge learned by SA nodes for local alignment.

Feature work

- Achituve, I., Maron, H. and Chechik, G., 2020. Self-Supervised Learning for Domain Adaptation on Point-Clouds. *arXiv preprint arXiv:2003.12641*.

Table 2: Accuracy per class (%)

Method	Bathtab	Bed	Bookshelf	Cabinet	Chair	Lamp	Monitor	Plant	Sofa	Table	Avg.
ModelNet to ScanNet											
# Samples	26	85	146	149	801	41	61	25	134	301	–
Unsupervised	48.7	41.2	40.9	**3.8**	54.1	29.3	**57.9**	82.7	43.0	28.8	43.0
PointDAN [30]	56.4	**61.5**	29.9	2.4	**71.7**	30	42.6	26.6	**53**	14.8	38.9
PCM + RegRec-T (ours)	**57.7**	41.2	**49.8**	2	59.8	**35**	53.6	**88**	47.5	**62.8**	**49.7**
ModelNet to ShapeNet											
# Samples	85	23	50	126	662	232	112	30	330	842	–
Unsupervised	81.2	17.4	96.7	**1.6**	89.4	**66.5**	84.5	86.7	90.6	88.8	70.3
PointDAN [30]	82	36.2	**97.3**	0	**94.6**	54.90	**93.50**	95.6	**92.9**	**91.5**	**73.8**
PCM + RegRec-T (ours)	**87.5**	**43.5**	**97.3**	1.1	92.6	48.7	89.6	**96.7**	90.9	89.3	**73.7**

Thank you for listening!