Metric Learning for 3D Point Clouds Using Optimal Transport

A. Implementation Details and Discussion

Dataset. In this section, we mention the implementation details of the training procedure and the different tasks that we perform. For working with ModelNet [2] and ShapeNet-Part [4], we randomly sample 2048 points to represent the object surface. For ScanObjectNN [1], we use the given pre-processed point cloud having 2048 points per object. We use the main split for all our experiments of ScanObjectNN provided by [1]. Here, an object which is represented with 2048 points can also have points that belong to the background. We do not use the mask label that indicates whether a point belongs to the foreground or background in any of our experiments. This interpretation is completely left to the network to decide and choose which points matter the most in order to classify a particular object. As a pre-processing step, we always normalize the coordinates of the point clouds and scale the object to bring it inside a unit sphere. Note that, the point clouds are not explicitly aligned across all datasets.

Contrastive learning. The fixed set of transformations used in contrastive learning are scaling, rotation, and noise jittering. The scaling parameter is randomly selected from a range of 0.8 to 1.25. For rotation, we only rotate the object about its Y-axis and choose a rotation angle between 0° to 360°. For points jittering, we first draw noise from a normal distribution having 0 mean and standard deviation as 0.01. This noise is then added to each point individually, introducing surface distortion. We compose these transformations by selecting two random input parameters for each of them, resulting in T_1 and T_2.

Architecture and method. We use a 3-layer MLP network as our encoder for all the classification and segmentation tasks. The number of neurons in each layer are $(64, 128, 256)$ with batch-norm and ReLU activation after every layer. Note that, the last layer's output is not passed through the activation function. After the max-pooling operation, the encoder gives a 256-dimensional vector, which is then converted into a discrete distribution z'_m depending on the selected ground metric space $(\mathbb{R}^2, \mathbb{R}^4, \mathbb{R}^8)$. For example, if the selected ground metric space is $\mathbb{R}^2$, then the 256-dimensional embedding vector is reshaped into $(256/2) \times 2$, giving us 128 support points in 2 dimensions. As we use a uniform distribution, each of the points have the same weightage of $(1/128)$, all effectively making it a discrete distribution in $\mathbb{R}^2$. We first pre-train the encoder using our defined $\mathcal{L}_{sup}$ or $\mathcal{L}_{self}$ with different distance measures. For Euclidean distances, we use the 256-dimensional vector straight away, and for the Wasserstein metric, we use the discrete representation. After pre-training, for evaluating on a test set of a particular dataset, we extract the learnt embeddings from the frozen pre-trained encoder and use a Linear SVM to perform classification. For the object segmentation task, we train a 3-layer MLP network to predict class labels for all points in a point cloud, where the input is the embeddings provided by the pre-trained encoder. Note that, the pre-trained encoder weights are frozen while training the segmentation network. The number of neurons in each layer are $(256, 128, 50)$, with batch-norm and ReLU after each layer except the last one. The global embedding and per-point embeddings are stacked and passed as input to the segmentation network.

We use FoldingNet [3] encoder and decoder for the interpolation task. For this task, the embedding size is 512. These encoder and decoder are trained with our defined $\mathcal{L}_{sup}$ that uses the discrete embeddings, together with chamfer distance as a reconstruction loss on the decoder's output. We use a weighted sum of these two losses as the total loss to backprop through the network. The weights assigned to $\mathcal{L}_{sup}$ and chamfer distance are 0.2 and 0.8 respectively. For Wasserstein metric, we use $\mathbb{R}^2$ as the ground metric dimension, and as a Euclidean metric, we use L^2-distance.

B. Explainability: Critical Points

We visualize and compare features captured by Wasserstein embeddings and Euclidean embeddings for more examples in Figure 1, 2. Our Wasserstein embeddings are able to capture the full skeleton structure of the given point cloud, whereas critical points captured by Euclidean embeddings are comparatively poor with uneven distribution and missing parts.

C. Ablation Study

We perform point perturbation and point density variation to test their effects on the encoders embeddings pre-trained with different distance metrics and report the clas-

sification accuracy on Modelnet10 and ScanObjectNN as shown in Figure 4, 5 respectively. We can see that our CL+SW_2 performs better in the robustness test for Modelnet10, but the margin of performance difference is less on ScanObjectNN.

D. 3D Shape Interpolation

We show more examples of 3D point cloud interpolations in Figure 3. For both Euclidean and Wasserstein embeddings, we use the 512 dimensional feature vector of the source and target sample and take the weighted sum of these embeddings to generate interpolated sample embedding. This interpolated embedding is then passed to the trained decoder to get a point cloud in 3D space. Overall, we can see smoothness in the generated point cloud surface (with better information) as well as in the embedding space itself, when the space is Wasserstein. In Table 1, we report the noise measure of the interpolated samples shown in Figure 3. Most of the interpolated samples from Wasserstein space have lower values of noise measures.

Table 1. Noise measure for interpolation results shown in Figure 3. Bold values represent smoother surfaces having less noise. The values are scaled by a factor of 10^3.

Figure	Method	Step 1	Step 2	Step 3	Step 4
3 (a)	Euclidean	31.5	36.8	**34.6**	**30.5**
	Wasserstein	**30.0**	**34.9**	35.0	31.8
3 (b)	Euclidean	29.8	31.8	33.4	**33.4**
	Wasserstein	**29.1**	**30.1**	**33.0**	34.5
3 (c)	Euclidean	**25.0**	**24.3**	**23.3**	**22.3**
	Wasserstein	25.3	24.4	**23.3**	22.5
3 (d)	Euclidean	24.9	26.2	26.0	**25.0**
	Wasserstein	**23.2**	**24.7**	**25.1**	25.3

References

[1] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019.

[2] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[3] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–215, 2018.

[4] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (TOG)*, 2016.

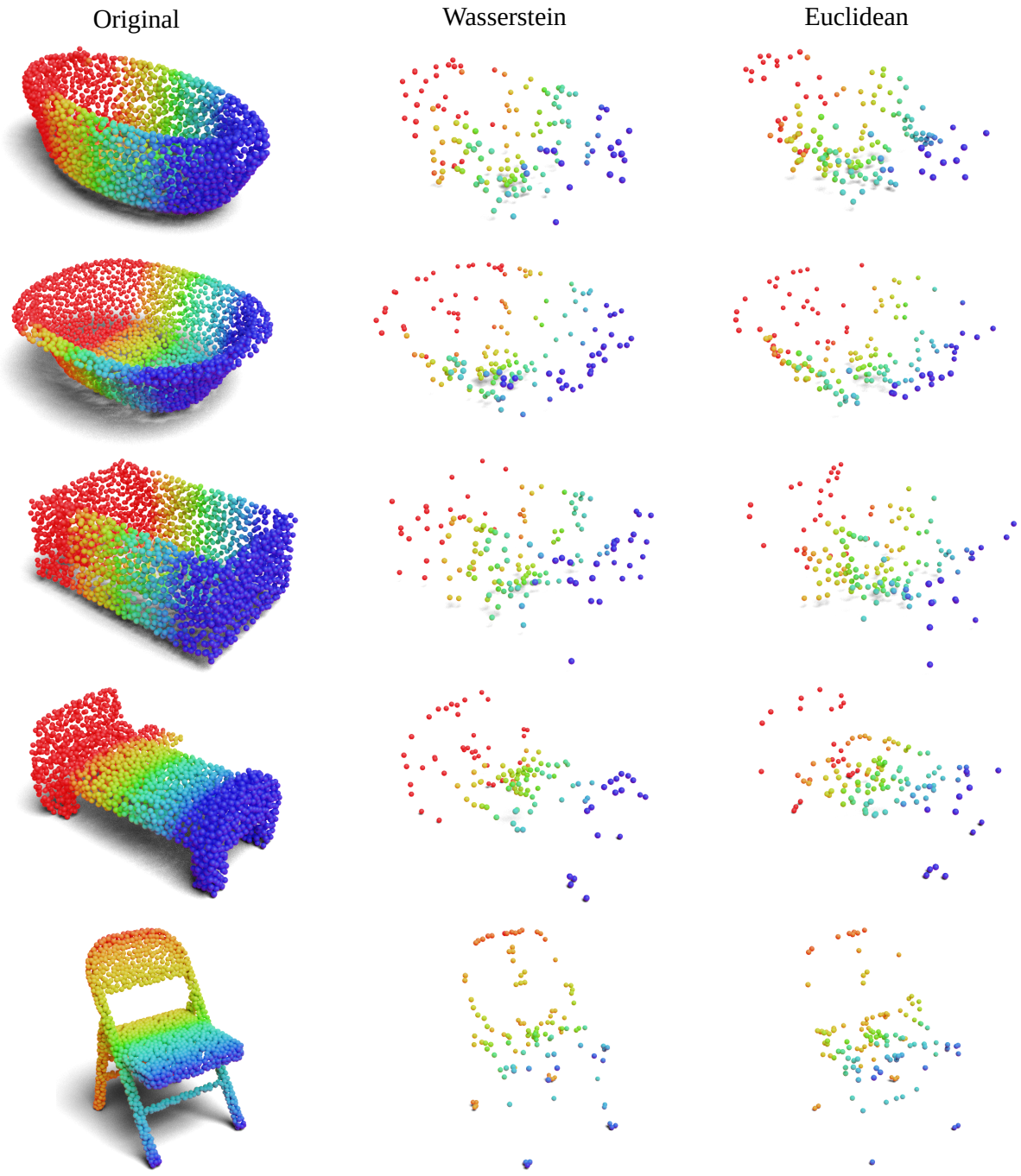


Figure 1. Comparison of important points given by network trained with Wasserstein and Euclidean metric. First column shows the original point cloud, second column shows the critical points set captured by Wasserstein space, and third column shows the same for Euclidean space.

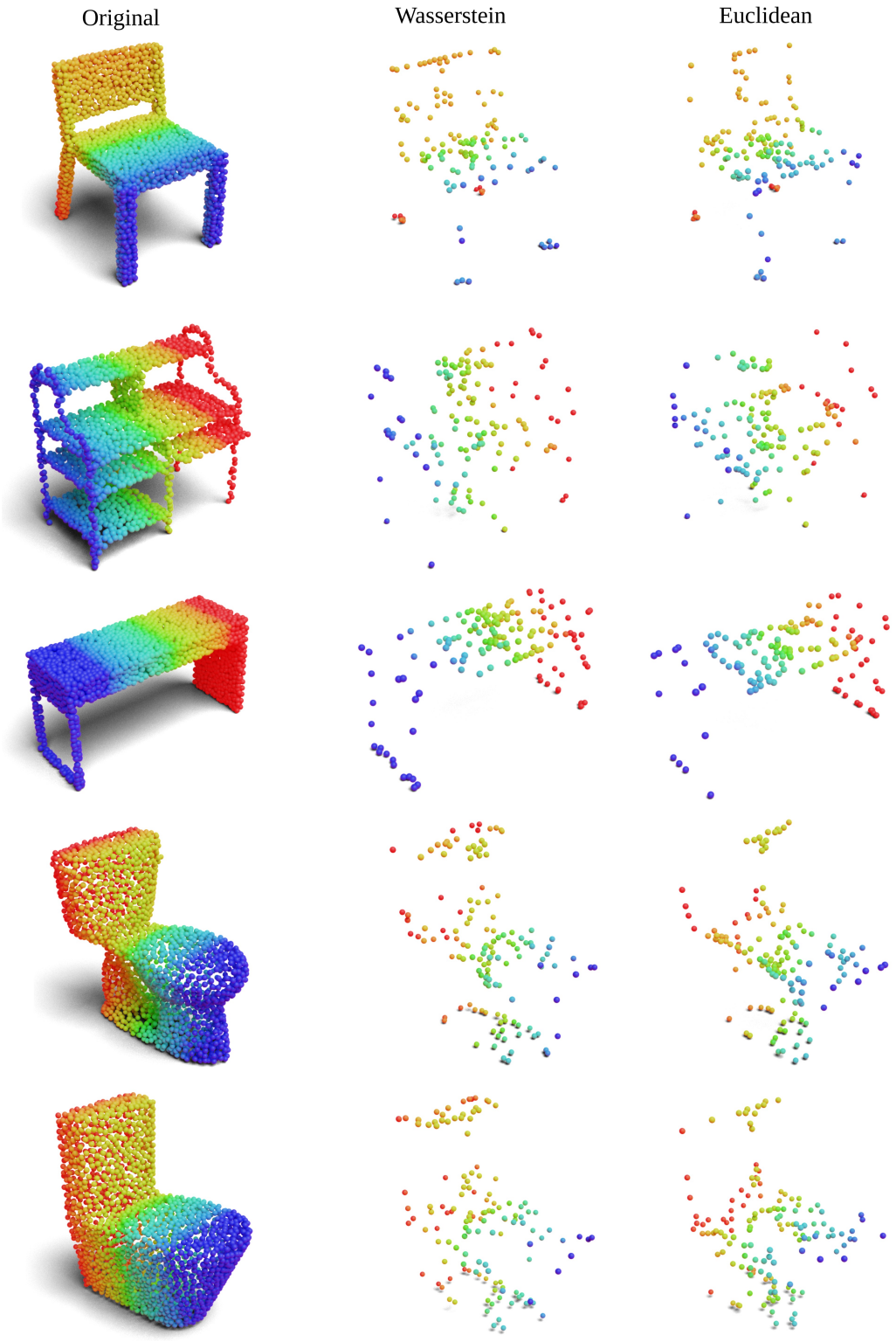


Figure 2. Comparison of important points given by network trained with Wasserstein and Euclidean metric. First column shows the original point cloud, second column shows the critical points set captured by Wasserstein space, and third column shows the same for Euclidean space.

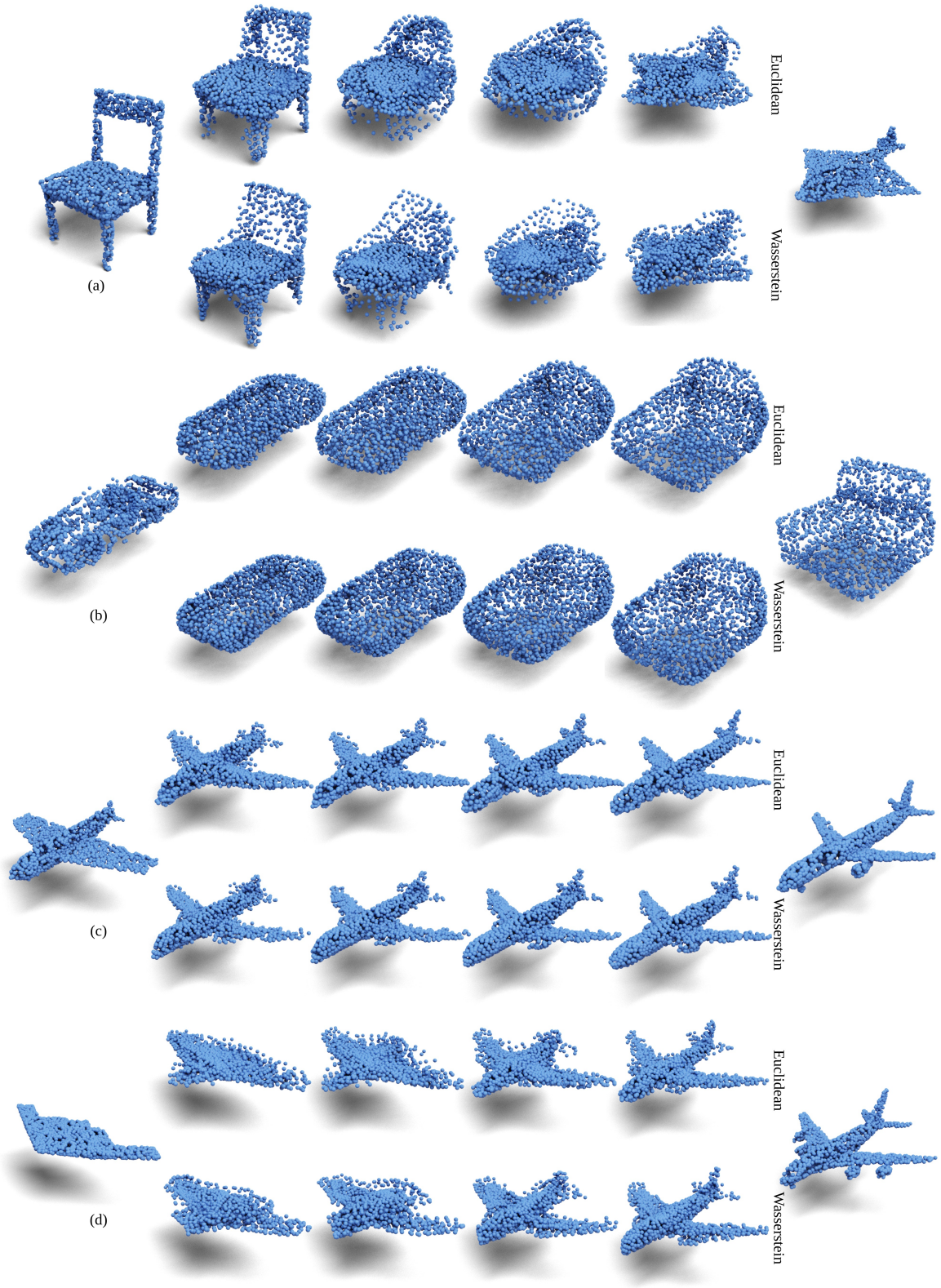
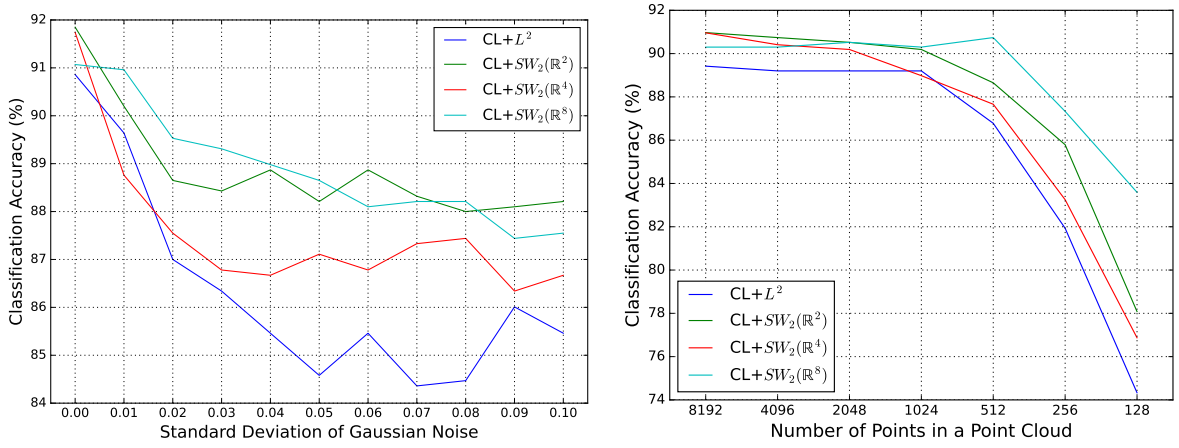
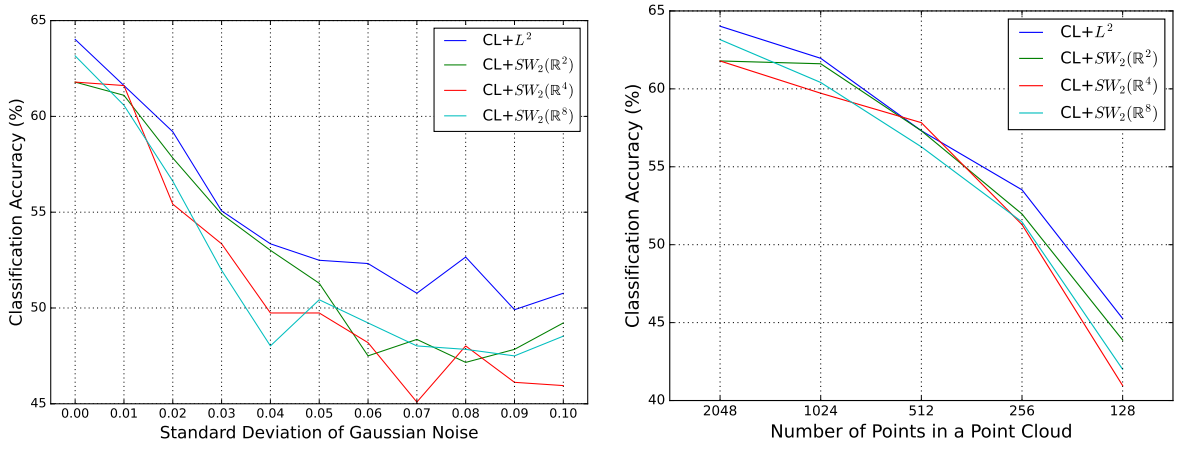


Figure 3. Comparision of interpolation from source (leftmost) to target (rightmost) samples between Euclidean embeddings and Wasserstein embeddings. For each sample, top row shows results from reconstruction after interpolating two point cloud embeddings in Euclidean space. Likewise, bottom row shows results obtained by Wasserstein embeddings. Weight ratio $(0.8, 0.6, 0.4, 0.2)$ (from left to right) with respect to the source.



(a) (b)

Figure 4. Robustness test with point perturbation and point sampling for our method CL+SW_2 and baseline CL+L_2 on Modelnet10.



(a) (b)

Figure 5. Robustness test with point perturbation and point sampling for our method CL+SW_2 and baseline CL+L_2 on ScanObjectNN.