Full Experiments

Here, we demonstrate the effectiveness of the fixed-RWB/free-RWB and the efficiency of the coreset technique. All the experiments were conducted on a server equipped with 2.40GHz Intel CPU, 128GB main memory, and Python 3.8.

We evaluate our method on three datasets: MNIST [1], ModelNet40 [4] and Human Connectome Project (HCP) [3].

MNIST: MNIST dataset [1] is a popular handwritten benchmark with digits from 0 to 9. We select 3000 images. For the l-th image, we represent it by a measure $\mu^l = \sum_{i=1}^{60} a_i^l \delta_{x_i^l} \in \mathcal{P}(\mathcal{X})$ via k-means clustering. More specifically, we take 28×28 pixels as the input of clustering, and obtain 60 cluster centers $X^l = \left\{x_i^l\right\}_{i\in[60]}$ as the locations of μ^l; the weight a_i^l is proportional to the total pixel number of this cluster. Till now, we obtain a clean dataset $\mathcal{Q} = \left\{\mu^l\right\}_{l\in[3000]}$. We assume $\mathcal{Q}'$ is its corresponding noisy dataset, which will be constructed for each dataset later.

ModelNet40: ModelNet40 [4] is a comprehensive clean collection of $3D$ CAD models. We choose 989 CAD models of chair. First, we convert these CAD models into point clouds. Then, each point cloud was grouped into $k = 60$ clusters; each cluster was represented by its cluster center; the weight of each center is proportional to the total number of points of the cluster. Then, we can obtain $\mathcal{Q}$ and $\mathcal{Q}'$ (as described in MNIST dataset).

Human Connectome Project (HCP): Human Connectome Project (HCP) [3] is a dataset of high-quality neuroimaging data in over 1100 healthy young adults, aged 22–35. We took 3000 3D brain images. For each image, its voxels were grouped into $k = 60$ clusters; each cluster was represented by its cluster center; the weight of each center is proportional to the total number of points of the cluster. Then, we can obtain $\mathcal{Q}$ and $\mathcal{Q}'$ (as described in MNIST dataset).

The method proposed by Le et al. [2] suffers from numerical instability caused by the KL-divergence regularization term. (This is illustrated in Appendix D.) In almost all the instances here, it failed to produce results. Thus, we only compare our method with the original Wasserstein barycenter (WB) algorithm here.[1]

To measure the performance of our methods, we consider three criteria: (i) running time: the CPU time consumed by algorithms; (ii) WD: the Wasserstein distance between the barycenter computed on the noisy dataset $\mathcal{Q}'$ and the WB on the clean dataset $\mathcal{Q}$; (iii) cost: Let ν be the barycenter computed by some algorithm on the noisy dataset $\mathcal{Q}'$. We define its cost as $WB(\mathcal{Q}, \nu)$. (Note that its cost is evaluated on the clean dataset $\mathcal{Q}$.) We run 10 trials and record the average results.

A Experiments on MNIST Dataset

This section shows the experimental result on MNIST dataset. First, to show the efficiency of our fixed-RWB/free-RWB, we add ζ mass of Gaussian noise to each measure $\mu^l \in \mathcal{Q}$ to obtain a noisy dataset $\mathcal{Q}'$. Then, we compute barycenter by the original fixed-support/free-support WB algorithm and our fixed-RWB/free-RWB on the noisy dataset respectively. The results in Table 1/Table 2 show that our fixed-RWB/free-RWB can tackle outliers effectively under different noise intensity.

Then, we show the efficiency of our coreset technique in Figures 1 to 9. In this scenario, to obtain a noisy dataset $\mathcal{Q}'$, we first add 0.1 mass of Gaussian noise from $\mathcal{N}(40, 40)$ to each measure $\mu^l \in \mathcal{Q}$, and then shift the locations of several images randomly according to a distribution $\mathcal{N}(0, \cdot)$. Throughout our experiments, we ensure that the total sample size of the uniform sampling method equals to the coreset size of our method. Γ is the sample size for each layer in our method. Our coreset method is much more time consuming. However, it performs well on the criteria WD and cost. Moreover, our method is more stable.

[1]The codes and full experiments (including the results on the other datasets and contrast experiments on the numerical instability issues [2]) are available at https://github.com/little-worm/iicclrr2023/blob/main/Full_experiments.pdf.

Table 1: Comparisons of our fixed-RWB and the original fixed-support WB algorithm under different noise intensity on MNIST. We use ζ to denote the total mass of outliers, and the noise distribution (N.D.) is Gaussian distribution $\mathcal{N}(\cdot,\cdot)$.

ζ	N.D.	Our fixed-RWB			WB		
		runtime ($\downarrow$)	WD ($\downarrow$)	cost ($\downarrow$)	runtime ($\downarrow$)	WD ($\downarrow$)	cost ($\downarrow$)
0.1	$\mathcal{N}(20,20^2)$	$89.38_{\pm 24.89}$	**0.14** $_{\pm 0.00}$	**3.54** $_{\pm 0.00}$	$99.11_{\pm 31.32}$	**0.14**$_{\pm 0.00}$	**3.54**$_{\pm 0.00}$
	$\mathcal{N}(40,40^2)$	$80.83_{\pm 23.43}$	**13.73** $_{\pm 0.10}$	**23.06** $_{\pm 0.11}$	$74.42_{\pm 13.06}$	$21.92_{\pm 0.23}$	$32.25_{\pm 0.26}$
	$\mathcal{N}(60,60^2)$	$66.78_{\pm 14.84}$	**88.49** $_{\pm 1.51}$	**86.17** $_{\pm 1.32}$	$75.15_{\pm 10.62}$	$189.53_{\pm 1.36}$	$175.33_{\pm 1.20}$
0.2	$\mathcal{N}(20,20^2)$	$67.96_{\pm 24.50}$	**0.29** $_{\pm 0.01}$	**4.52** $_{\pm 0.00}$	$96.60_{\pm 24.53}$	$0.30_{\pm 0.01}$	$4.53_{\pm 0.00}$
	$\mathcal{N}(40,40^2)$	$100.53_{\pm 25.94}$	**25.42** $_{\pm 0.19}$	**38.18** $_{\pm 0.22}$	$119.50_{\pm 30.14}$	$44.04_{\pm 0.24}$	$60.19_{\pm 0.29}$
	$\mathcal{N}(60,60^2)$	$77.53_{\pm 17.41}$	**171.08**$_{\pm 1.65}$	**165.06**$_{\pm 1.53}$	$90.63_{\pm 19.26}$	$386.12_{\pm 2.05}$	$365.86_{\pm 1.92}$
0.3	$\mathcal{N}(20,20^2)$	$85.55_{\pm 18.26}$	**0.57** $_{\pm 0.01}$	**5.25** $_{\pm 0.00}$	$106.25_{\pm 38.59}$	$0.58_{\pm 0.01}$	$5.26_{\pm 0.00}$
	$\mathcal{N}(40,40^2)$	$103.50_{\pm 27.60}$	**27.50** $_{\pm 0.18}$	**41.44** $_{\pm 0.21}$	$102.17_{\pm 34.68}$	$69.02_{\pm 0.35}$	$89.88_{\pm 0.42}$
	$\mathcal{N}(60,60^2)$	$98.17_{\pm 17.41}$	**190.22**$_{\pm 1.72}$	**197.55**$_{\pm 1.76}$	$68.74_{\pm 14.46}$	$552.93_{\pm 2.79}$	$563.60_{\pm 2.82}$

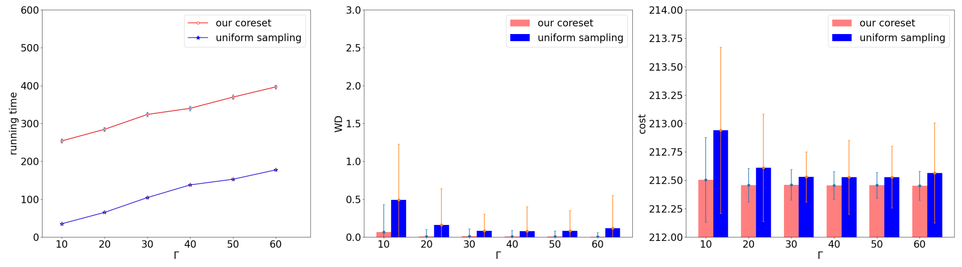

Figure 1: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 100 images according to distribution $\mathcal{N}(0, 40)$.

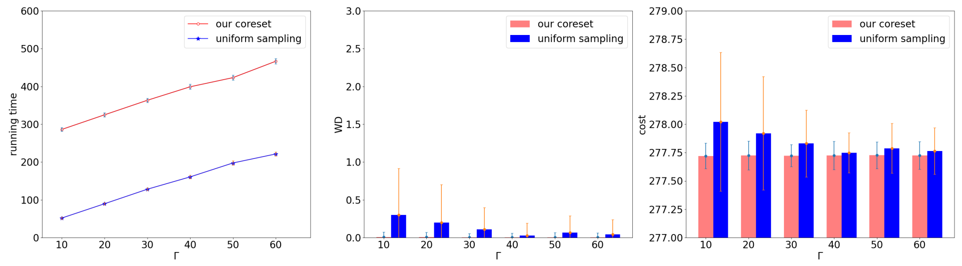

Figure 2: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 200 images according to distribution $\mathcal{N}(0, 40)$.

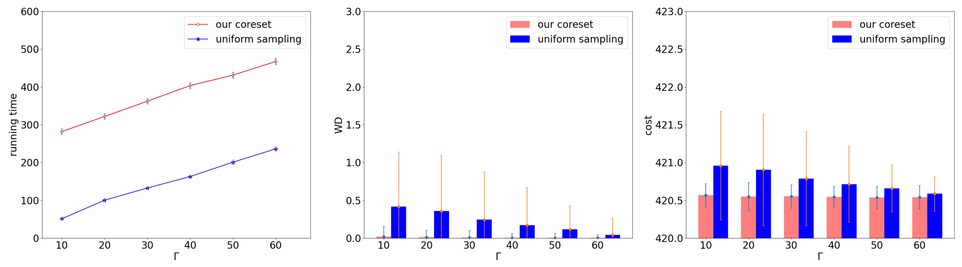

Figure 3: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 300 images according to distribution $\mathcal{N}(0, 40)$.

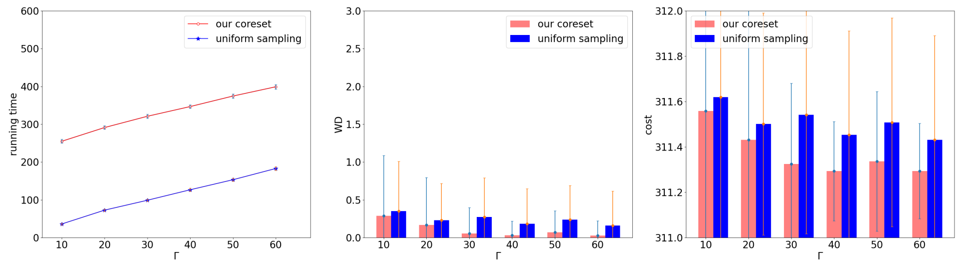

Figure 4: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 100 images according to distribution $\mathcal{N}(0, 60)$.

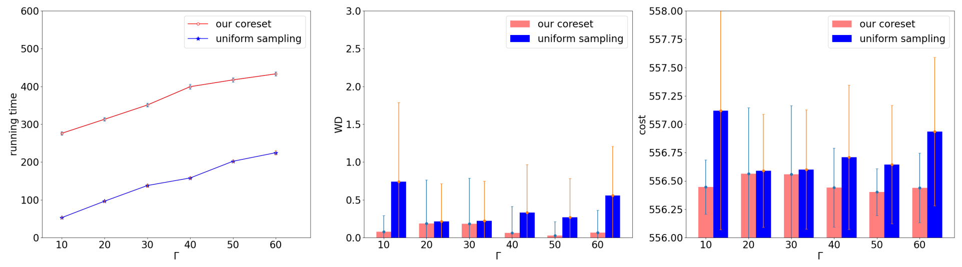

Figure 5: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 200 images according to distribution $\mathcal{N}(0, 60)$.

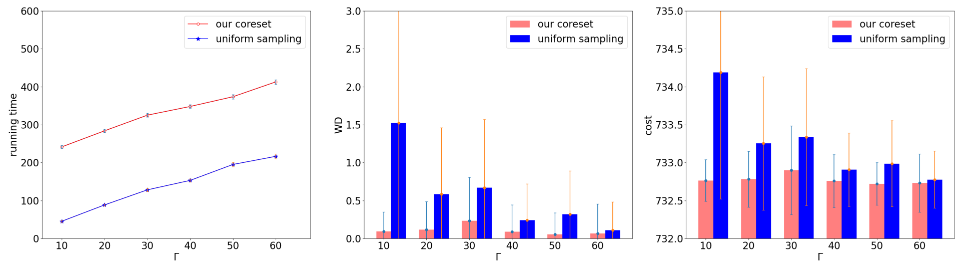

Figure 6: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 300 images according to distribution $\mathcal{N}(0, 60)$.

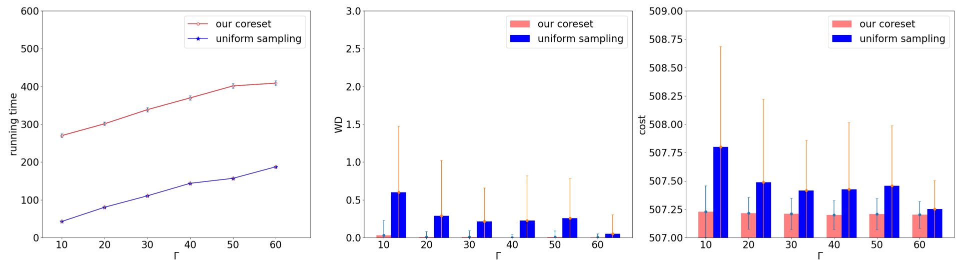
Figure 7: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 100 images according to distribution $\mathcal{N}(0, 80)$.

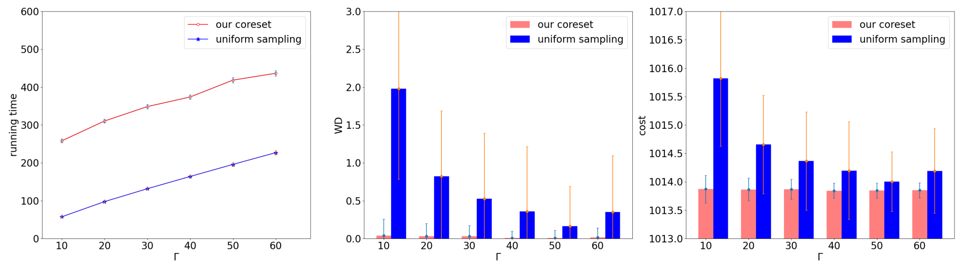
Figure 8: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 200 images according to distribution $\mathcal{N}(0, 80)$.

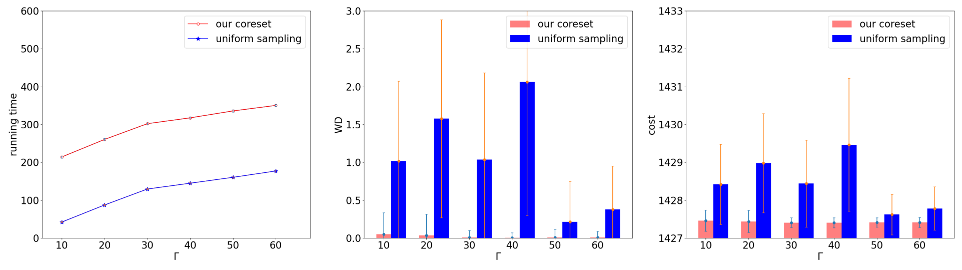
Figure 9: Comparisons of our coreset and uniform sampling on MNIST. Shift locations of 300 images according to distribution $\mathcal{N}(0, 80)$.

Table 2: Comparisons of our free-RWB and the original free-support WB algorithm under different noise intensity on MNIST. We use ζ to denote the total mass of outliers, and the noise distribution (N.D.) is Gaussian distribution $\mathcal{N}(\cdot,\cdot)$.

ζ	N.D.	Our free-RWB			WB		
		runtime ($\downarrow$)	WD ($\downarrow$)	cost ($\downarrow$)	runtime ($\downarrow$)	WD ($\downarrow$)	cost ($\downarrow$)
0.1	$\mathcal{N}(20,20^2)$	$406.26_{\pm 101.29}$	$\mathbf{0.56}_{\pm 0.02}$	$\mathbf{57.98}_{\pm 0.02}$	$429.72_{\pm 125.86}$	$0.57_{\pm 0.02}$	$57.99_{\pm 0.02}$
	$\mathcal{N}(40,40^2)$	$374.55_{\pm 111.53}$	$\mathbf{2.86}_{\pm 0.11}$	$\mathbf{59.46}_{\pm 0.11}$	$371.07_{\pm 109.45}$	$91.27_{\pm 31.51}$	$114.97_{\pm 27.88}$
	$\mathcal{N}(60,60^2)$	$299.06_{\pm 29.56}$	$\mathbf{0.85}_{\pm 0.04}$	$\mathbf{58.27}_{\pm 0.04}$	$313.12_{\pm 42.11}$	$473.66_{\pm 7.86}$	$444.91_{\pm 6.86}$
0.2	$\mathcal{N}(20,20^2)$	$434.46_{\pm 71.85}$	$\mathbf{2.50}_{\pm 0.09}$	$\mathbf{59.47}_{\pm 0.09}$	$465.89_{\pm 105.96}$	$2.56_{\pm 0.09}$	$59.52_{\pm 0.09}$
	$\mathcal{N}(40,40^2)$	$462.63_{\pm 109.26}$	$\mathbf{15.78}_{\pm 3.13}$	$\mathbf{59.81}_{\pm 1.01}$	$448.58_{\pm 67.21}$	$382.56_{\pm 6.81}$	$335.22_{\pm 6.47}$
	$\mathcal{N}(60,60^2)$	$460.43_{\pm 66.94}$	$\mathbf{3.71}_{\pm 0.12}$	$\mathbf{59.22}_{\pm 0.26}$	$481.08_{\pm 56.36}$	$1201.98_{\pm 8.23}$	$1054.11_{\pm 7.66}$
0.3	$\mathcal{N}(20,20^2)$	$477.99_{\pm 74.03}$	$\mathbf{5.84}_{\pm 0.11}$	$\mathbf{61.41}_{\pm 0.20}$	$509.11_{\pm 48.46}$	$5.99_{\pm 0.11}$	$61.51_{\pm 0.21}$
	$\mathcal{N}(40,40^2)$	$306.84_{\pm 39.58}$	$\mathbf{198.63}_{\pm 74.20}$	$\mathbf{174.58}_{\pm 62.32}$	$312.85_{\pm 35.12}$	$661.14_{\pm 7.50}$	$540.72_{\pm 7.90}$
	$\mathcal{N}(60,60^2)$	$444.65_{\pm 110.59}$	$\mathbf{10.86}_{\pm 1.14}$	$\mathbf{56.03}_{\pm 0.76}$	$519.71_{\pm 76.13}$	$2093.29_{\pm 10.88}$	$1811.10_{\pm 9.60}$

B Experiments on HCP Dataset

This section shows the experimental result on HCP dataset. First, to show the efficiency of our fixed-RWB/free-RWB, we add ζ mass of Gaussian noise to each measure $\mu^l \in \mathcal{Q}$ to obtain a noisy dataset $\mathcal{Q}'$. Then, we compute barycenter by the original fixed-support/free-support WB algorithm and our fixed-RWB/free-RWB on the noisy dataset respectively. The results in Table 3/Table 4 show that our fixed-RWB/free-RWB can tackle outliers effectively under different noise intensity.

Table 3: Comparisons of our fixed-RWB and the original fixed-support WB algorithm under different noise intensity on HCP. We use ζ to denote the total mass of outliers, and the noise distribution (N.D.) is Gaussian distribution $\mathcal{N}(\cdot,\cdot)$.

ζ	N.D.	Our fixed-RWB			WB		
		runtime ($\downarrow$)	WD ($\downarrow$)	cost ($\downarrow$)	runtime ($\downarrow$)	WD ($\downarrow$)	cost ($\downarrow$)
0.1	$\mathcal{N}(50,50^2)$	$101.50_{\pm 35.27}$	$\mathbf{4.98}_{\pm 0.12}$	$97.39_{\pm 0.05}$	$92.71_{\pm 25.00}$	$5.20_{\pm 0.11}$	$\mathbf{97.34}_{0.05}$
	$\mathcal{N}(100,100^2)$	$84.28_{\pm 29.66}$	$\mathbf{89.45}_{\pm 1.39}$	$\mathbf{179.87}_{\pm 1.37}$	$105.27_{\pm 25.10}$	$321.63_{\pm 1.72}$	$404.51_{1.67}$
	$\mathcal{N}(150,150^2)$	$83.36_{\pm 6.68}$	$\mathbf{82.40}_{\pm 3.90}$	$\mathbf{176.28}_{\pm 3.94}$	$89.49_{\pm 24.22}$	$2039.19_{\pm 7.51}$	$2134.79_{7.46}$
0.2	$\mathcal{N}(50,50^2)$	$61.77_{\pm 16.90}$	$\mathbf{10.15}_{\pm 0.23}$	$96.36_{\pm 0.05}$	$65.90_{\pm 16.08}$	$10.78_{\pm 0.24}$	$\mathbf{96.24}_{0.05}$
	$\mathcal{N}(100,100^2)$	$77.48_{\pm 17.16}$	$\mathbf{169.68}_{\pm 1.28}$	$\mathbf{256.04}_{\pm 1.29}$	$72.97_{\pm 18.30}$	$718.88_{\pm 3.54}$	$788.35_{3.40}$
	$\mathcal{N}(150,150^2)$	$52.19_{\pm 3.50}$	$\mathbf{231.62}_{\pm 5.89}$	$\mathbf{327.27}_{\pm 6.05}$	$49.86_{\pm 1.99}$	$4505.75_{\pm 18.61}$	$4584.84_{18.32}$
0.3	$\mathcal{N}(50,50^2)$	$95.91_{\pm 33.15}$	$\mathbf{15.40}_{\pm 0.09}$	$95.66_{\pm 0.04}$	$107.73_{\pm 22.91}$	$16.60_{\pm 0.10}$	$\mathbf{95.55}_{0.05}$
	$\mathcal{N}(100,100^2)$	$93.63_{\pm 27.43}$	$\mathbf{184.54}_{\pm 2.30}$	$\mathbf{268.10}_{\pm 2.34}$	$99.77_{\pm 25.78}$	$1180.03_{\pm 6.22}$	$1227.38_{6.04}$
	$\mathcal{N}(150,150^2)$	$49.84_{\pm 3.15}$	$\mathbf{422.77}_{\pm 6.48}$	$\mathbf{521.64}_{\pm 6.48}$	$50.87_{\pm 5.31}$	$7182.62_{\pm 16.07}$	$7211.00_{15.77}$

Then, we show the efficiency of our coreset technique in Figures 10 to 18. In this scenario, to obtain a noisy dataset $\mathcal{Q}'$, we first add 0.1 mass of Gaussian noise from $\mathcal{N}(40,40)$ to each measure $\mu^l \in \mathcal{Q}$, and then shift the locations of several images randomly according to a distribution $\mathcal{N}(0,\cdot)$. Throughout our experiments, we ensure that the total sample size of the uniform sampling method equals to the coreset size of our method. Γ is the sample size for each layer in our method. Our coreset method is much more time consuming. However, it performs well on the criteria WD and cost. Moreover, our method is more stable. (The results of our coreset method perform well with high probability, so it is reasonable that our method performs worse than uniform sampling with small probability.)

Table 4: Comparisons of our free-RWB and the original free-support WB algorithm under different noise intensity on HCP. We use ζ to denote the total mass of outliers, and the noise distribution (N.D.) is Gaussian distribution $\mathcal{N}(\cdot,\cdot)$.

ζ	N.D.	Our free-RWB			WB		
		runtime (↓)	WD (↓)	cost (↓)	runtime (↓)	WD (↓)	cost (↓)
0.1	$\mathcal{N}(50, 50^2)$	$364.40_{\pm 55.35}$	$\mathbf{2.79}_{\pm 0.10}$	$924.84_{\pm 0.13}$	$391.94_{\pm 46.67}$	$3.19_{\pm 0.10}$	$\mathbf{924.63}_{\pm 0.17}$
	$\mathcal{N}(100, 100^2)$	$362.76_{\pm 41.85}$	$\mathbf{2.53}_{\pm 0.08}$	$\mathbf{925.81}_{\pm 0.16}$	$393.62_{\pm 51.49}$	$1428.06_{\pm 53.70}$	$1758.54_{\pm 36.74}$
	$\mathcal{N}(150, 150^2)$	$377.66_{\pm 68.12}$	$\mathbf{0.09}_{\pm 0.01}$	$\mathbf{927.67}_{\pm 0.03}$	$390.97_{\pm 72.66}$	$4912.27_{\pm 52.62}$	$4581.39_{\pm 48.59}$
0.2	$\mathcal{N}(50, 50^2)$	$385.24_{\pm 74.60}$	$\mathbf{11.78}_{\pm 0.56}$	$910.16_{\pm 0.74}$	$402.37_{\pm 74.39}$	$14.10_{\pm 0.62}$	$\mathbf{906.11}_{\pm 0.72}$
	$\mathcal{N}(100, 100^2)$	$392.26_{\pm 73.57}$	$\mathbf{9.96}_{\pm 0.32}$	$\mathbf{918.17}_{\pm 1.02}$	$385.29_{\pm 50.78}$	$3962.80_{\pm 179.01}$	$3409.19_{\pm 147.24}$
	$\mathcal{N}(150, 150^2)$	$347.45_{\pm 27.78}$	$\mathbf{0.30}_{\pm 0.02}$	$\mathbf{927.10}_{\pm 0.06}$	$400.30_{\pm 25.87}$	$12873.93_{\pm 88.97}$	$11040.97_{\pm 78.38}$
0.3	$\mathcal{N}(50, 50^2)$	$342.88_{\pm 24.33}$	$\mathbf{30.53}_{\pm 0.50}$	$867.25_{\pm 1.57}$	$377.99_{\pm 45.68}$	$38.19_{\pm 0.77}$	$\mathbf{847.05}_{\pm 2.33}$
	$\mathcal{N}(100, 100^2)$	$347.99_{\pm 43.53}$	$\mathbf{30.51}_{\pm 7.15}$	$\mathbf{889.46}_{\pm 4.19}$	$400.05_{\pm 40.38}$	$7414.53_{\pm 51.70}$	$5896.24_{\pm 53.92}$
	$\mathcal{N}(150, 150^2)$	$43.53_{\pm 56.00}$	$\mathbf{7.15}_{\pm 0.03}$	$\mathbf{4.19}_{\pm 0.09}$	$40.38_{\pm 26.55}$	$51.70_{\pm 87.81}$	$53.92_{\pm 76.20}$

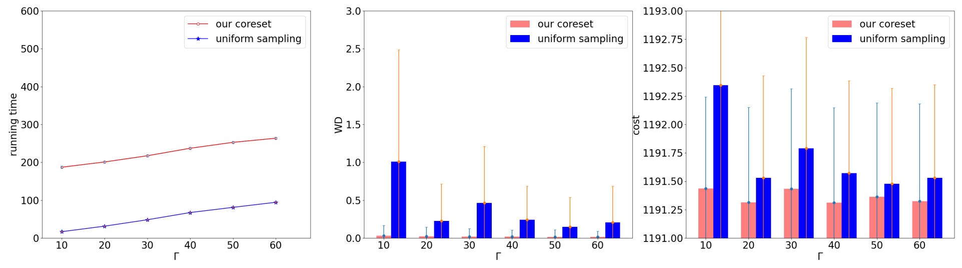

Figure 10: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 100 brains according to distribution $\mathcal{N}(0, 50)$.

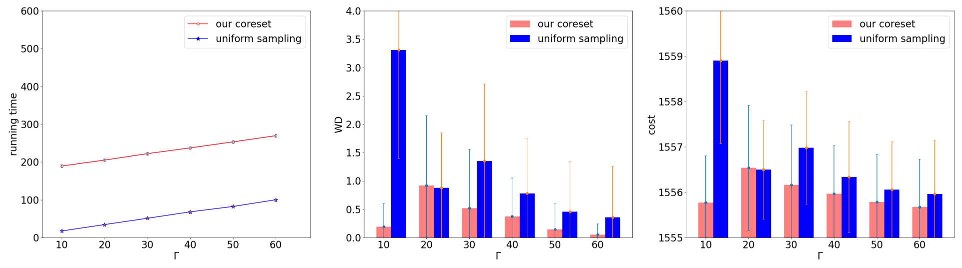

Figure 11: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 200 brains according to distribution $\mathcal{N}(0, 50)$.

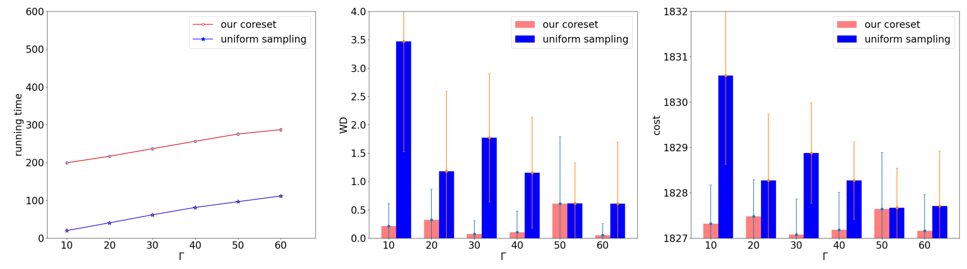

Figure 12: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 300 brains according to distribution $\mathcal{N}(0, 50)$.

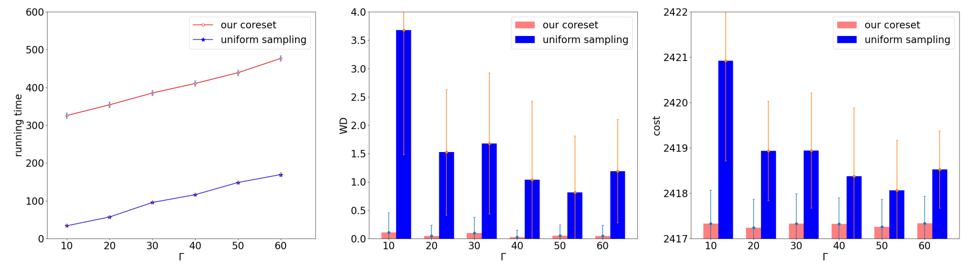

Figure 13: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 100 brains according to distribution $\mathcal{N}(0, 100)$.

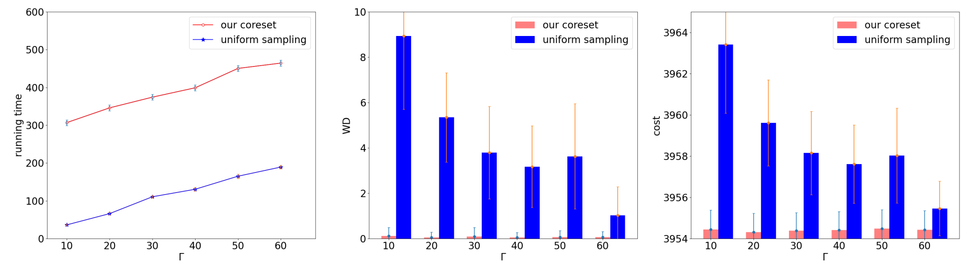

Figure 14: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 200 brains according to distribution $\mathcal{N}(0, 100)$.

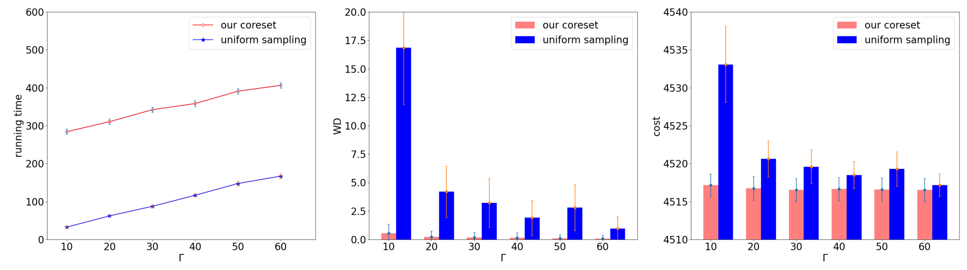

Figure 15: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 300 brains according to distribution $\mathcal{N}(0, 100)$.

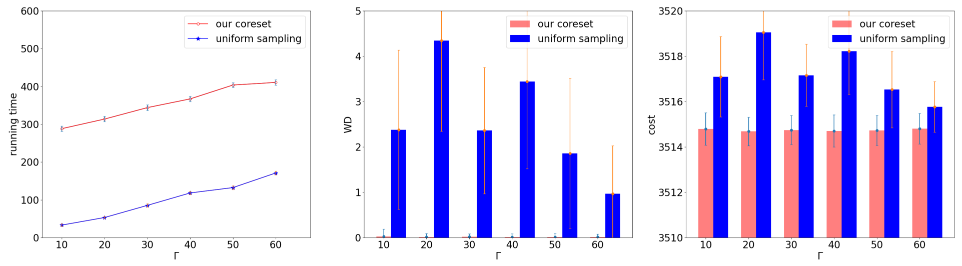

Figure 16: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 100 brains according to distribution $\mathcal{N}(0, 150)$.

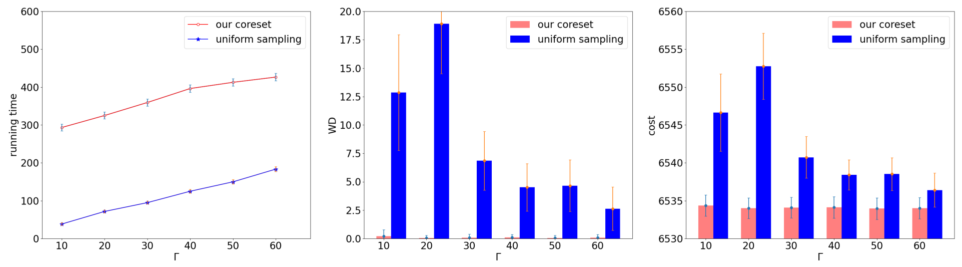

Figure 17: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 200 brains according to distribution $\mathcal{N}(0, 150)$.

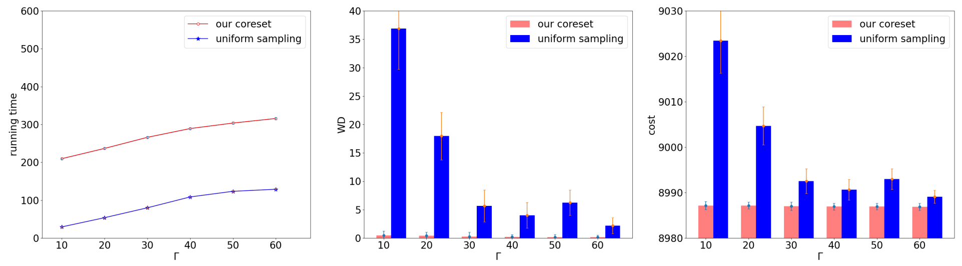

Figure 18: Comparisons of our coreset and uniform sampling on HCP. Shift locations of 300 brains according to distribution $\mathcal{N}(0, 150)$.

C Experiments on ModelNet40 Dataset

This section shows the experimental result on ModelNet40 dataset. First, to show the efficiency of our fixed-RWB/free-RWB, we add ζ mass of Gaussian noise to each measure $\mu^l \in \mathcal{Q}$ to obtain a noisy dataset $\mathcal{Q}'$. Then, we compute barycenter by the original fixed-support/free-support WB algorithm and our fixed-RWB/free-RWB on the noisy dataset respectively. The results in Table 5/Table 6 show that our fixed-RWB/free-RWB can tackle outliers effectively under different noise intensity. In our experiments, we can only obtain a local approximate solution, the solution is somewhat random. Thus, it is reasonable that our method obtain worse result when the noise intensity is small.

Table 5: Comparisons of our fixed-RWB and the original fixed-support WB algorithm under different noise intensity on ModelNet40. We use ζ to denote the total mass of outliers, and the noise distribution (N.D.) is Gaussian distribution $\mathcal{N}(\cdot,\cdot)$.

ζ	N.D.	Our fixed-RWB			WB		
		runtime (↓)	WD (↓)	cost (↓)	runtime (↓)	WD (↓)	cost (↓)
0.1	$\mathcal{N}(50, 50^2)$	32.63$_{\pm 11.52}$	**73.57** $_{\pm 0.58}$	**187.15** $_{\pm 0.48}$	34.98$_{\pm 9.08}$	79.87$_{\pm 0.73}$	192.85$_{\pm 0.56}$
	$\mathcal{N}(100, 100^2)$	10.72$_{\pm 43.08}$	**7.87** $_{\pm 572.59}$	**8.08** $_{\pm 697.28}$	36.17$_{\pm 37.89}$	1172.51$_{\pm 3438.73}$	1278.68$_{\pm 3560.87}$
	$\mathcal{N}(150, 150^2)$	31.16$_{\pm 5.56}$	**172.03** $_{\pm 1.14}$	**259.79** $_{\pm 1.09}$	32.70$_{\pm 7.09}$	194.93$_{\pm 1.53}$	280.03$_{\pm 1.46}$
0.2	$\mathcal{N}(50, 50^2)$	32.37$_{\pm 6.97}$	**1141.13**$_{\pm 11.48}$	**1242.84**$_{\pm 11.41}$	33.31$_{\pm 6.10}$	2677.97$_{\pm 9.06}$	2760.90$_{\pm 8.88}$
	$\mathcal{N}(100, 100^2)$	38.34$_{\pm 9.18}$	**1456.09**$_{\pm 24.07}$	**1583.98**$_{\pm 24.01}$	37.31$_{\pm 11.37}$	7526.14$_{\pm 50.15}$	7638.26$_{\pm 50.09}$
	$\mathcal{N}(150, 150^2)$	32.30$_{\pm 8.32}$	**276.55** $_{\pm 3.00}$	**344.85** $_{\pm 2.82}$	32.48$_{\pm 8.99}$	324.69$_{\pm 3.40}$	388.14$_{\pm 3.08}$
0.3	$\mathcal{N}(50, 50^2)$	40.42$_{\pm 8.82}$	**1368.73**$_{\pm 16.04}$	**1465.01**$_{\pm 16.19}$	43.17$_{\pm 10.26}$	4314.01$_{\pm 25.69}$	4335.73$_{\pm 24.59}$
	$\mathcal{N}(100, 100^2)$	23.20$_{\pm 1.56}$	**2283.81**$_{\pm 41.22}$	**2402.69**$_{\pm 41.07}$	21.27$_{\pm 1.38}$	12023.66$_{\pm 31.14}$	12025.97$_{\pm 30.12}$
	$\mathcal{N}(150, 150^2)$	38.18$_{\pm 11.32}$	**2279.04**$_{\pm 27.75}$	**2398.06**$_{\pm 27.65}$	35.47$_{\pm 6.61}$	12032.18$_{\pm 40.15}$	12034.24$_{\pm 38.74}$

Table 6: Comparisons of our free-RWB and the original free-support WB algorithm under different noise intensity on ModelNet40. We use ζ to denote the total mass of outliers, and the noise distribution (N.D.) is Gaussian distribution $\mathcal{N}(\cdot,\cdot)$.

ζ	N.D.	Our free-RWB			WB		
		runtime (↓)	WD (↓)	cost (↓)	runtime (↓)	WD (↓)	cost (↓)
0.1	$\mathcal{N}(50, 50^2)$	165.92$_{\pm 25.89}$	**161.05**$_{\pm 85.74}$	**700.58**$_{\pm 37.09}$	160.59$_{\pm 14.37}$	305.74$_{\pm 28.49}$	759.37$_{\pm 22.66}$
	$\mathcal{N}(100, 100^2)$	147.06$_{\pm 18.13}$	**5.17** $_{\pm 0.31}$	**694.28**$_{\pm 1.88}$	154.87$_{\pm 14.97}$	2727.47$_{\pm 63.60}$	2524.60$_{\pm 54.60}$
	$\mathcal{N}(150, 150^2)$	162.39$_{\pm 12.54}$	**0.34** $_{\pm 0.05}$	**697.43**$_{\pm 0.13}$	159.90$_{\pm 14.85}$	6485.18$_{\pm 98.90}$	5754.88$_{\pm 81.64}$
0.2	$\mathcal{N}(50, 50^2)$	147.41$_{\pm 25.81}$	1047.55$_{\pm 42.76}$	1143.70$_{\pm 32.79}$	148.08$_{\pm 23.28}$	**1036.94**$_{\pm 27.18}$	**1132.61**$_{\pm 22.44}$
	$\mathcal{N}(100, 100^2)$	133.21$_{\pm 10.56}$	**472.29**$_{\pm 507.73}$	**918.99**$_{\pm 306.73}$	147.36$_{\pm 10.85}$	6725.00$_{\pm 75.05}$	5554.06$_{\pm 60.25}$
	$\mathcal{N}(150, 150^2)$	156.13$_{\pm 21.13}$	**2.06** $_{\pm 0.18}$	**697.74**$_{\pm 0.34}$	178.76$_{\pm 27.98}$	16143.00$_{\pm 226.85}$	13870.42$_{\pm 199.02}$
0.3	$\mathcal{N}(50, 50^2)$	162.40$_{\pm 13.90}$	1899.46$_{\pm 46.76}$	1651.95$_{\pm 35.70}$	161.80$_{\pm 21.29}$	**1836.57**$_{\pm 35.69}$	**1604.22**$_{\pm 28.64}$
	$\mathcal{N}(100, 100^2)$	160.68$_{\pm 13.73}$	**2498.13**$_{\pm 751.33}$	**2245.65**$_{\pm 543.33}$	203.94$_{\pm 20.47}$	11554.34$_{\pm 149.35}$	9428.57$_{\pm 131.31}$
	$\mathcal{N}(150, 150^2)$	157.10$_{\pm 24.86}$	**5.05** $_{\pm 0.30}$	**698.02** $_{\pm 0.61}$	211.04$_{\pm 30.49}$	26774.86$_{\pm 243.22}$	23069.52$_{\pm 222.73}$

Then, we show the efficiency of our coreset technique in Figures 19 to 27. In this scenario, to obtain a noisy dataset $\mathcal{Q}'$, we first add 0.1 mass of Gaussian noise from $\mathcal{N}(40, 40)$ to each measure $\mu^l \in \mathcal{Q}$, and then shift the locations of several images randomly according to a distribution $\mathcal{N}(0,\cdot)$. Throughout our experiments, we ensure that the total sample size of the uniform sampling method equals to the coreset size of our method. Γ is the sample size for each layer in our method. Our coreset method is much more time consuming. However, it performs well on the criteria WD and cost. Moreover, our method is more stable. (The results of our coreset method perform well with high probability, so it is reasonable that our method performs worse than uniform sampling with small probability.)

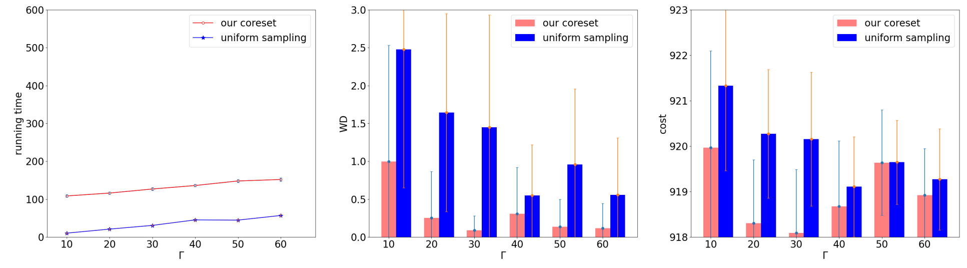

Figure 19: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 30 CAD models according to distribution $\mathcal{N}(0, 50)$.

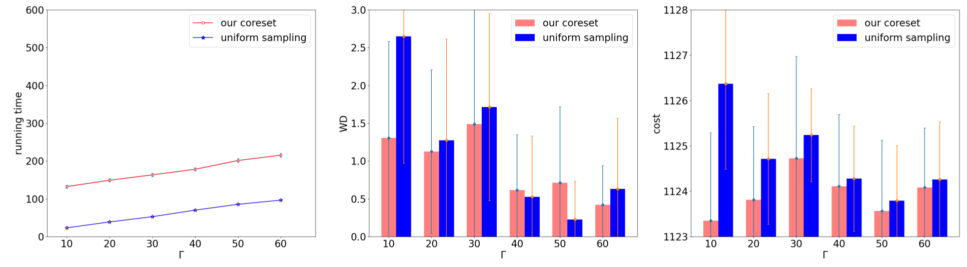

Figure 20: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 60 CAD models according to distribution $\mathcal{N}(0, 50)$.

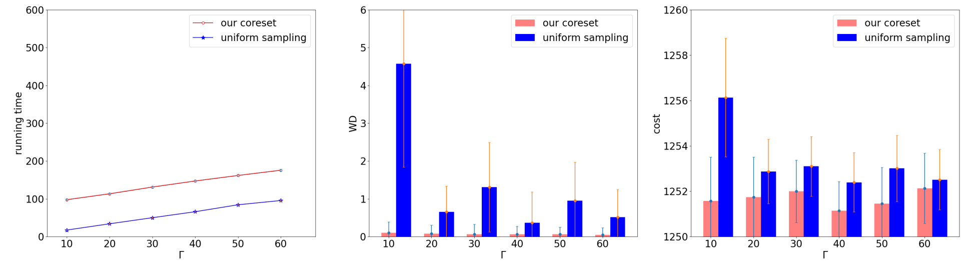

Figure 21: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 90 CAD models according to distribution $\mathcal{N}(0, 50)$.

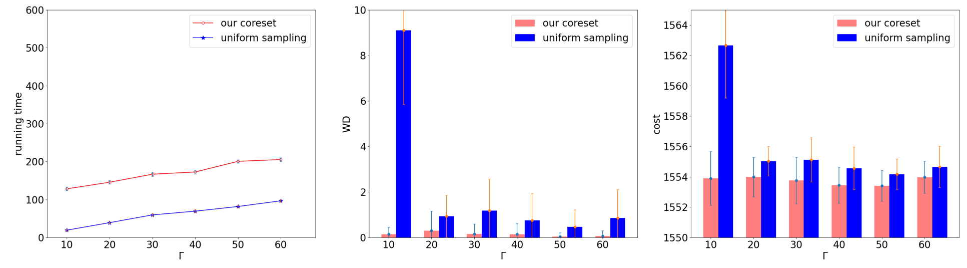

Figure 22: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 30 CAD models according to distribution $\mathcal{N}(0, 100)$.

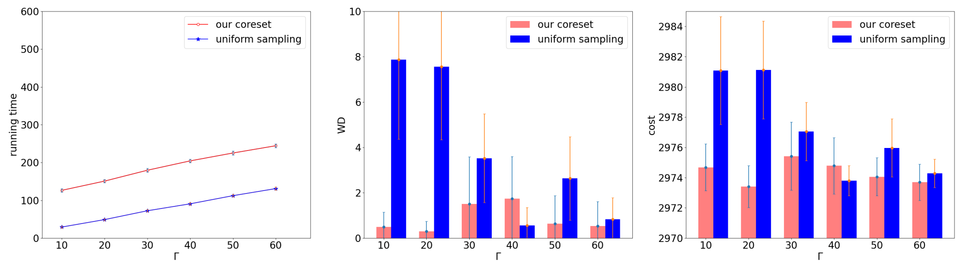

Figure 23: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 60 CAD models according to distribution $\mathcal{N}(0, 100)$.

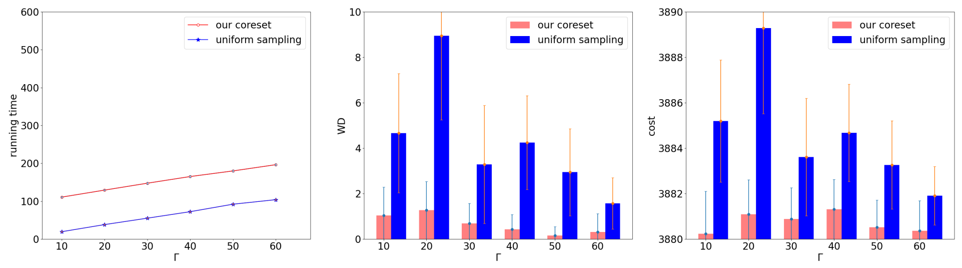

Figure 24: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 90 CAD models according to distribution $\mathcal{N}(0, 100)$.

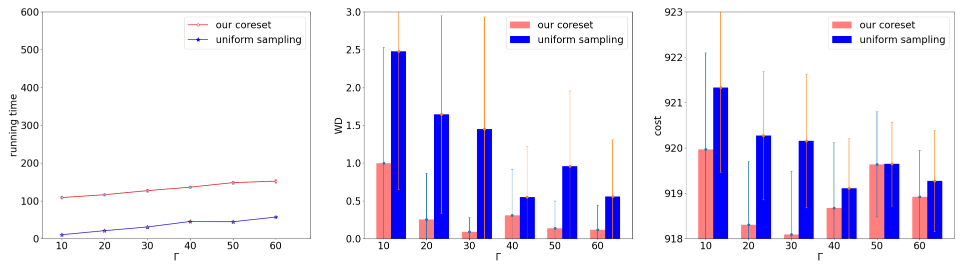

Figure 25: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 30 CAD models according to distribution $\mathcal{N}(0, 150)$.

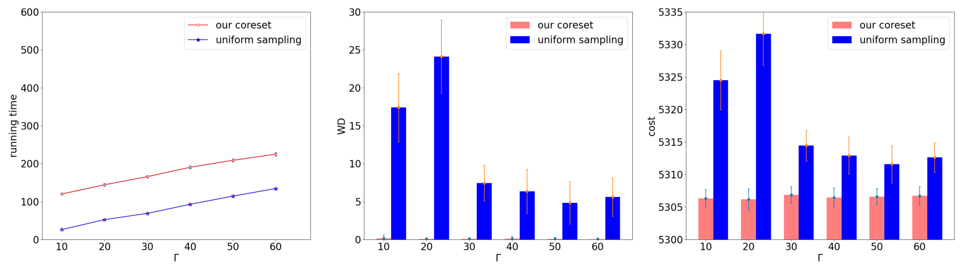

Figure 26: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 60 CAD models according to distribution $\mathcal{N}(0, 150)$.

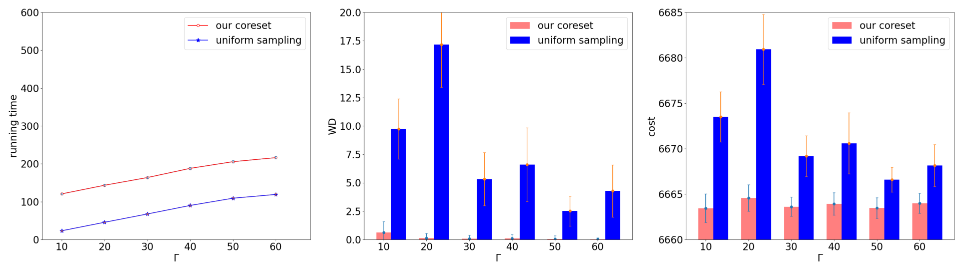

Figure 27: Comparisons of our coreset and uniform sampling on ModelNet40. Shift locations of 90 CAD models according to distribution $\mathcal{N}(0, 150)$.

D Numerical Instability of Le et al. [2]

In this section, we perform experiments on MNIST dataset to illustrate the numerical instability of Le et al. [2]. We select 300 images and obtain a clean dataset $\mathcal{Q}$. We add $\zeta = 0.2$ mass of Gaussian noise from $\mathcal{N}(40, \cdot)$ to each measure $\mu^l \in \mathcal{Q}$ to obtain a noisy dataset $\mathcal{Q}'$. Then, we compute barycenter by the original fixed-support WB algorithm and our fixed-RWB on the noisy dataset respectively. The results in the following table show that the method in [2] suffers from numerical instability. The symbol # denotes that we cannot compute the result due to numerical instability problem.

N.D.	method	runtime (↓)	WD (↓)	cost (↓)
$\mathcal{N}(40, 0^2)$	our fixed-RWB	5.06	178.99	181.66
	fixed-support WB	5.23	179.62	182.29
	UOT ($\gamma = 1$)	2.34	332.58	331.17
	UOT ($\gamma = 4$)	2.34	246.93	248.12
	UOT ($\gamma = 16$)	2.34	198.82	199.93
	UOT ($\gamma = 64$)	2.33	181.89	182.59
	UOT ($\gamma = 256$)	2.33	177.15	177.75
	UOT ($\gamma = 1024$)	2.51	175.93	176.50
$\mathcal{N}(40, 5^2)$	our fixed-RWB	4.61	170.53	181.79
	fixed-support WB	4.24	172.74	184.04
	UOT ($\gamma = 1$)	2.29	6.55	9.49
	UOT ($\gamma = 4$)	2.17	32.63	37.17
	UOT ($\gamma = 16$)	2.18	201.32	212.21
	UOT ($\gamma = 64$)	2.25	379.00	390.25
	UOT ($\gamma = 256$)	2.23	452.12	463.62
	UOT ($\gamma = 1024$)	2.22	474.26	485.96
$\mathcal{N}(40, 10^2)$	our fixed-RWB	5.19	132.14	136.37
	fixed-support WB	3.42	157.34	161.61
	UOT ($\gamma = 1$)	#	#	#
	UOT ($\gamma = 4$)	#	#	#
	UOT ($\gamma = 16$)	#	#	#
	UOT ($\gamma = 64$)	#	#	#
	UOT ($\gamma = 256$)	#	#	#

Continued on next page

– **continued from previous page**

N.D.	method	runtime (↓)	WD (↓)	cost (↓)
	UOT ($\gamma = 1024$)	#	#	#
$\mathcal{N}(40, 15^2)$	our fixed-RWB	6.81	60.66	72.76
	fixed-support WB	3.51	124.52	139.82
	UOT ($\gamma = 1$)	#	#	#
	UOT ($\gamma = 4$)	#	#	#
	UOT ($\gamma = 16$)	#	#	#
	UOT ($\gamma = 64$)	#	#	#
	UOT ($\gamma = 256$)	#	#	#
	UOT ($\gamma = 1024$)	#	#	#
$\mathcal{N}(40, 20^2)$	our fixed-RWB	7.57	48.39	50.94
	fixed-support WB	3.66	130.31	129.04
	UOT ($\gamma = 1$)	#	#	#
	UOT ($\gamma = 4$)	#	#	#
	UOT ($\gamma = 16$)	#	#	#
	UOT ($\gamma = 64$)	#	#	#
	UOT ($\gamma = 256$)	#	#	#
	UOT ($\gamma = 1024$)	#	#	#
$\mathcal{N}(40, 25^2)$	our fixed-RWB	4.18	39.79	42.23
	fixed-support WB	4.37	114.63	116.23
	UOT ($\gamma = 1$)	#	#	#
	UOT ($\gamma = 4$)	#	#	#
	UOT ($\gamma = 16$)	#	#	#
	UOT ($\gamma = 64$)	#	#	#
	UOT ($\gamma = 256$)	#	#	#
	UOT ($\gamma = 1024$)	#	#	#
$\mathcal{N}(40, 30^2)$	our fixed-RWB	7.59	35.06	35.82
	fixed-support WB	8.75	127.58	112.89
	UOT ($\gamma = 1$)	#	#	#
	UOT ($\gamma = 4$)	#	#	#
	UOT ($\gamma = 16$)	#	#	#
	UOT ($\gamma = 64$)	#	#	#
	UOT ($\gamma = 256$)	#	#	#
	UOT ($\gamma = 1024$)	#	#	#
$\mathcal{N}(40, 35^2)$	our fixed-RWB	8.83	20.71	24.00
	fixed-support WB	6.61	100.41	103.10
	UOT ($\gamma = 1$)	#	#	#
	UOT ($\gamma = 4$)	#	#	#
	UOT ($\gamma = 16$)	#	#	#
	UOT ($\gamma = 64$)	#	#	#
	UOT ($\gamma = 256$)	#	#	#
	UOT ($\gamma = 1024$)	#	#	#
$\mathcal{N}(40, 40^2)$	our fixed-RWB	5.13	12.55	16.49
	fixed-support WB	6.22	108.43	104.04
	UOT ($\gamma = 1$)	#	#	#
	UOT ($\gamma = 4$)	#	#	#
	UOT ($\gamma = 16$)	#	#	#
	UOT ($\gamma = 64$)	#	#	#
	UOT ($\gamma = 256$)	#	#	#
	UOT ($\gamma = 1024$)	#	#	#

Acknowledgment "Data were provided (in part) by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University."

References

[1] *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[2] K. Le, H. Nguyen, Q. M. Nguyen, T. Pham, H. Bui, and N. Ho, *On robust optimal transport: Computational complexity and barycenter computation*, Advances in Neural Information Processing Systems, 34 (2021), pp. 21947–21959.

[3] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium, et al., *The wu-minn human connectome project: an overview*, Neuroimage, 80 (2013), pp. 62–79.

[4] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, *3d shapenets: A deep representation for volumetric shapes*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1912–1920.