LocalSampleNet

Project Number: 20-1-1-2095

Submitted By

Amit Shomer 203961412

Royi Avron 302261813

Advisors

Prof. Shai Avidan Tel Aviv University

Itai Lang Tel Aviv University

Location:

Tel Aviv University

Contents

List of Tables

List of Figures

List of Equation

List of used Shortcuts

LSN – LocalSampleNet

SN – SampleNet

FPS - Farthest Point Sampling

MLP - Multilayer perceptron

FC- Fully connected

KNN - K-nearest neighbors

S&G -Sample & Group

MD- ModelNet

SR- Sample Ratio

Abstract

The popularity of 3D sensing devices like LiDAR and Stereo increased in the recent years. Point cloud is a set of points, produced by these devices, represents a visual scene, and can be used for different tasks. By sampling a smaller number of points, it is possible to reduce cost and process time and still complete the task with good performances.

LocalSampleNet purpose is to extend SampleNet [1], a point cloud sampling Neural network, by incorporating local information in the sampling process. It is based on PointNet++ [2] a neural network architecture which introduced an hierarchical model that extracts local features from a small geometric structure neighborhood. By localizing the sampling network, we achieved better generalization ability for point clouds structures, compared to the global sampling of SampleNet (SN).

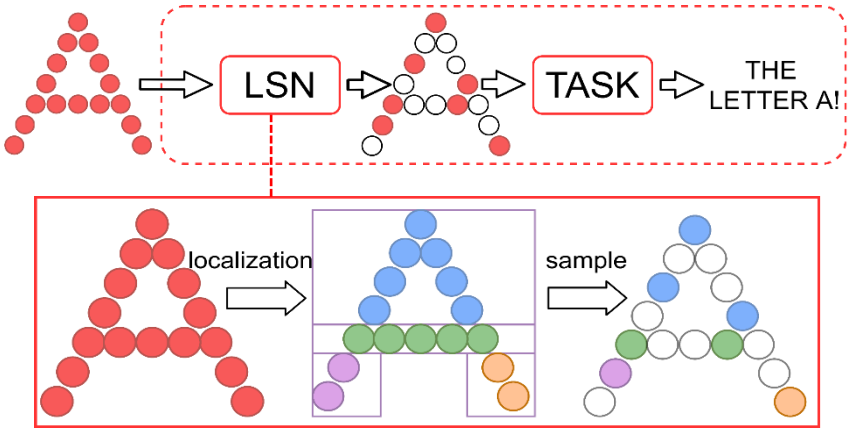


Figure 1 High level local sampling block diagram

First, the input data is divided into local region areas by sample points from original point cloud and gathering neighborhood points to patches. From each patch the network extracts local information and produce simplified points from it. Those points are fed to the task network, in our case - classification. Local SampleNet achieves better generalization compared to SampleNet. For example, SampleNet accuracy is higher by 6.8% at sample ratio 32 on structures it was not trained on (ModelNet30). Another variant of LocalSampleNet reached very good accuracy of 88.02% for structures it was trained on (MD10) with sample ratio 32.

1 Introduction

3D sensing has an increased popularity in recent years. These sensors capture data which can be represented in a variety of ways, one of them is a point cloud. Point cloud is a set of 3D points (x, y, z) that capture information from a 3D scene. Point clouds have several unique properties:

- **Unordered**. Unlike pixel array in images, point cloud is invariant to permutation if it's members.
- **Neighborhood.** As point clouds represent a geometric structure, nearby local structure and interactions among local points is significant.

Multiple tasks, such as registration, classification, and reconstruction, process point clouds as their input information. Although it is possible to process the whole point cloud, by sampling it, it is possible to minimize cost and process time while maintaining good performances. For example, SampleNet [1] showed that while using its architecture with a sample Ratio of 32, they achieved a 90% computation reduction, with 80% accuracy and only increased memory usage by 6%.

Sampling can be done in a variety of ways. The naive approach is to reduce data by a random selection points from the input data. A more efficient and common method is the **Farthest point sampling** (FPS). FPS starts with a selection of a random point in a set of size N, and iteratively selects the point f_i that has the largest geodesic distance from the points that have already been selected $f_0 \dots f_{i-1}$, until f_n sampled points are chosen. *FPS* method supplies a good spatial coverage of the input point cloud [3][4]. However, those are non-learned sampling methods.

Deep Learning approaches showed in the last few years a phenomenal success in processing point cloud for different applications. Learning a sampling method, as in "SampleNet" and "Learning to Sample" [6], showed high performance compared to non-learning sampling methods.

The objective of our project is to extend the work of SampleNet and train a neural network to sample point clouds while preserving local information. By localizing the sampling network, we aim to achieve better localization ability compared to SampleNet. Our approach combines the benefits of the SampleNet approach with the concept of partition sets of points into local regions, as was introduced in *PointNet++* [2].

2 Related Work and Theoretical Background

The First Neural network whose operated directly on unordered point cloud data is the work of Qi et al. *PointNet* [5]. Qi introduced a cutting age method dealing with point clouds – *MLP*: multi-layer perceptron. This concept allows the network to transform the input data from the coordinate space *i.e.* $[x, y, z]$, to a more complex feature vector space through the max pooling method. This feature vector can then be furthered processed to the task in hand (*e.g.* classification). The *PointNet* classifier was taken as our task. As the *MLP* concept layers.

The work of Lang *et al. "SampleNet"* and the work of Dovrat *et al. "Learning to Sample"* [6], showed that using a learning method of sampling, based on MLP concept, it is possible to achieve high performance results while sampling a lower number of points, compared to other sampling techniques. From these works, we've used the simplify and projection concepts. For Clarity we will briefly explain the mention concepts taken from the papers. [1] We will begin by define the flowing, that can be viewed in Figure 2

N Input data set

S Simplified set

P Projected set

2.1 Simplify

Dovrat *et al.* [4], showed a concept of separating the sampling network from the task one. For the sampling network, they showed a Simplification of point cloud: By reducing an Input data cloud of Size N to size S, where the latter is ideal for the task and its points are near to those of N. The satisfy the second demand, two losses are defined:

$$\mathcal{L}_a(X, Y) = \frac{1}{|X|}\sum_{x \in X} \min\left||x - y|\right|_2^2$$

Equation 1 Average nearest neighbor loss

$$\mathcal{L}_m(X, Y) = \max \min\left||x - y|\right|_2^2$$

Equation 2 Maximal nearest neighbor loss

Thus, the simplification Loss is defined as:

$$\mathcal{L}_{simplify}(S, N) = \mathcal{L}_a(S, N) + \beta\mathcal{L}_m(S, N) + (\gamma + \delta|S|)\mathcal{L}_a(N, S)$$

[1] A more detailed explanation may be reviewed in the SampleNet and Dovrat *et al* papers.

2.2 Project

Using the simplification concept of point clouds suggested above, SampleNet 0 furthered the sampling process and introduced a projection concept of sampling. The soft Projection operation donates the following concept: each point $s \in S$ is projected on its neighborhood, defined by its k nearest neighbors (KNN) in N, to produce a set of projected points p.

Each p is defined by:

$$p = \sum_{i \in \mathcal{N}_N(s)} \omega_i n_i$$

Equation 3 p is the weighted average of original points

$$\omega_i = \frac{e^{-d_i^2/t^2}}{\sum_{i \in \mathcal{N}_N(q)} e^{-d_i^2/t^2}}$$

Equation 4 the weighted coefficient of p

Where $\mathcal{N}_{N(s)}$ are the indices of the KKK of S in N, and t is a learnable temperature factor, $d_i \equiv \left|\left|s - n_i\right|\right|_2$. It is important to note, that in the for $t \to 0$ Equation 3 converges to the $Kronecker\ delta\ function$, placed at the nearest neighbor. henceforth to achieve the above, the *projection loss* is given by:

$$\mathcal{L}_{project} = t^2$$

Equation 5 Projection Loss

The task is then fed with the set of P points assuming that each $p \in P$ approximates the optimal subset $P^* \in N$ for the task under deliberation. It is important to note, that in evaluation, instead of soft projection, the task receives a sampled $P^* \in N$, such as each $p^* \epsilon P^*$ is selected by the point n_i with the highest projection weight. The Total $Loss$ for our network is the same as in SampleNet 0:

$$\mathcal{L}_{total}^{samp} = \mathcal{L}_{task}(P) + \alpha \mathcal{L}_{simplify}(S, N) + \lambda \mathcal{L}_{project}$$

Equation 6 SampleNet Loss [1]

PointNet++ [2], a neural network that extracts local features from a small geometric structure neighborhood, may be observed as an expanding of PointNet [5]. It is composed of a hierarchical grouping structure: Divided to a set of levels that are composed of a *sampling & Grouping* layer

and *PointNet* layer. Each level produces less elements to the following one. This concept will be furthered elaborated at 3.1 as it was used in our method.

3 Method and implementation

An overview of our *LSN* architecture is described in Figure 2. First, the classifier task was pre-trained on ModelNet40 dataset and its wights were frozen. Afterwards *LSN* trained on an input size N, in our case 1024, from ModelNet10 dataset (subset data of ModelNet40), and down sampled it to a smaller set of size P, that was fed to the task. We examined each implementation with sample ratio from 1 to 128 compared to the original input data.

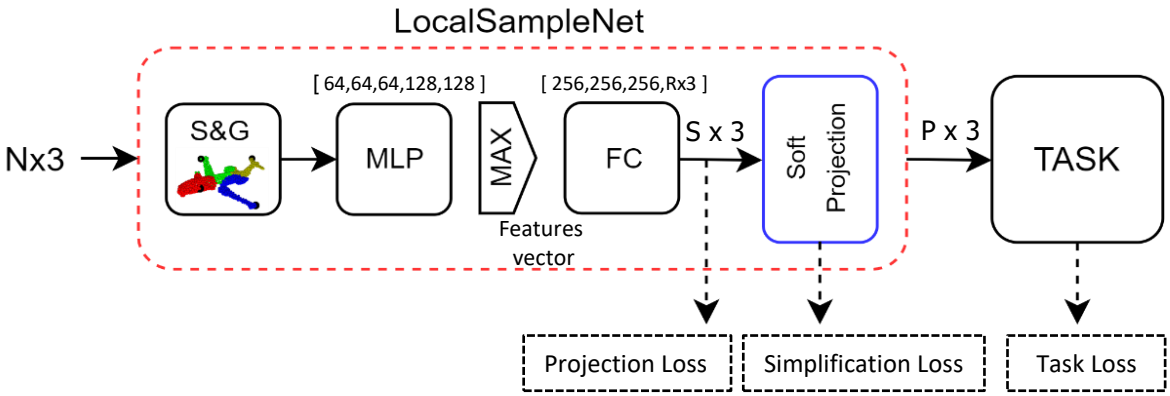


Figure 2 LSN Block Diagram

All MLP and Fully Connected layers are followed by ReLU and batch-normalization layer except for the output layer.

3.1 Sample & Group

The ***Sampling layer***: Given a point cloud input a subset set of points; *seeds*, were chosen, via *FPS* method by default. Learning sample method was implemented and will present in section 4.4 SampleSeeds.

The ***Grouping layer*****:** from each *seed* a local region was defined; the k nearest neighborhood (*KNN*) around each seed were chosen. *KNN* is defined by metric distance. At the end of this process the cloud is divided to D sub local regions, termed *Patches*. Each *patch* had been shifted and normalized before procced further through the *MLP* layers. The process can be examined in Figure 3 bellow.

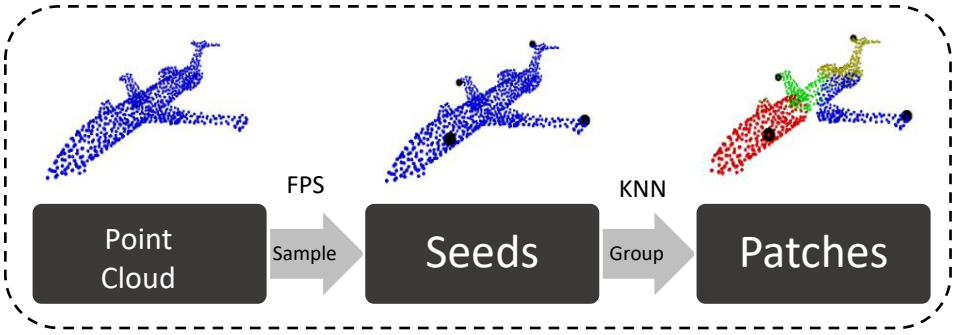


Figure 3 *Sample & Group flow. Number of patches and points per patch were chosen to maximize results.*

3.2 Features vector

Each patch was processed through to the *MLP* layers flowed by *max pooling* to generate $D \times 128$ *Feature Vectors*. We implanted two ways to process these *Feature Vectors:*

In the first Method all *D Feature vectors* were forwarded to the *Fully Connected* (FC) layers in parallel, downsizing each *patch* to a set of simplified local points. Each *patch* contributed an equal number of N/D , later concatenated the simplified $S \times 3$ point cloud.

To overcome the restriction which each patch contributed the same amount of points, we tried second method, termed ***concat feature vector***. In this implementation after the *MLP* and max pooling the *feature vectors* were concatenated to one feature vector of size $(128 \cdot D) \times 1$. The concat feature vector was forwarded through the *FC* layers and downsized to a simplified $S \times 3$ point cloud.

In the evaluation process, we choose the closest points from the original point cloud to each simplified point, and those points were fed to the task.

4 Experiments and Results

LocalSampleNet was evaluated on the disjoint sets $MD10$, $MD30$ which are subsets of $MD40$ [7]. Each point cloud contained 1024 points that were uniformly sample from the dataset models. We used the official train test split for MD10 and MD40.

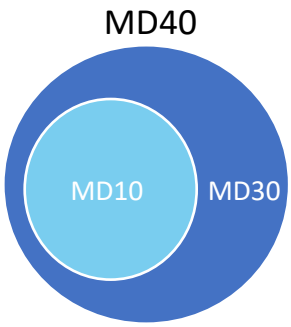


Figure 4 ***Split of data.*** ModelNet40 (MD40) contain 9843 train sets and 2468 test sets. MD10 and MD30 is disjoint to each other and substes for MD40. MD10 contain 3991 data set and 908 test data. MD30 contain 1560 test data**.**

4.1 Number of Patches and points per patch

The first experiment for each architecture was to determine the optimal number of patches and number of points per patch. This experiment was done with a *SR* of 32. After finding the optimal values we procced with the same configuration to the rest Sample ratios. We can view in Table 1 evaluation of LSN concat features vector architecture, and Table 2 evaluation of LSN.

Patch Number	**Points per patch**	**MD10**	**MD30**	**MD40**
2	512	87.44	25.38	48.34
4	256	86.78	25.44	47.12
8	64	82.71	25	41.95
	128	**88.02**	**28.39**	**49.39**
	256	87.99	24.93	49.31
16	64	87.33	24.87	47.12
32	32	83.48	25.89	47.36

Table 1 ***LocalSampleNet concat features vector results*** *for sample ratio 32. After this experiments all sample ratio were evaluated with 8 patches and 128 points per patch*

Patch Number	**Points per patch**	**MD10**	**MD30**	**MD40**
2	512	48.89	23.7	30.4
4	256	56.16	25.32	36.14
8	128	58.37	27.82	38.61
16	64	57.71	29.61	39.42
32	32	**72.06**	**36.622**	**48.62**

Table 2 ***LocalSampleNet results*** *for sample ratio 32. After this experiments all sample ratio were evaluated with 32 patches and 32 points per patch.*

4.2 Sample Method Comparison

We compared **LSN** and **LSN concat feature vector** performance for classification task against FPS sampling and SN. MD10 LSN concat feature vector maintains a high performance for different sample ratios. For example, at SR 32 LSN concat feature vector accuracy is 88.02% which is only 1.6% lower than SampleNet

LSN **concat feature vector** was trained and evaluate with 8 patch and 128 points per patch. From Sample ratio 1 to 32 LocalSampleNet was trained and evaluate on 32 patches and 32 points per patch. Sample ratio 64 was divided to 16 patch and 64 point per patch and for sample ratio 128 we used 8 patch and 128 points per patch.

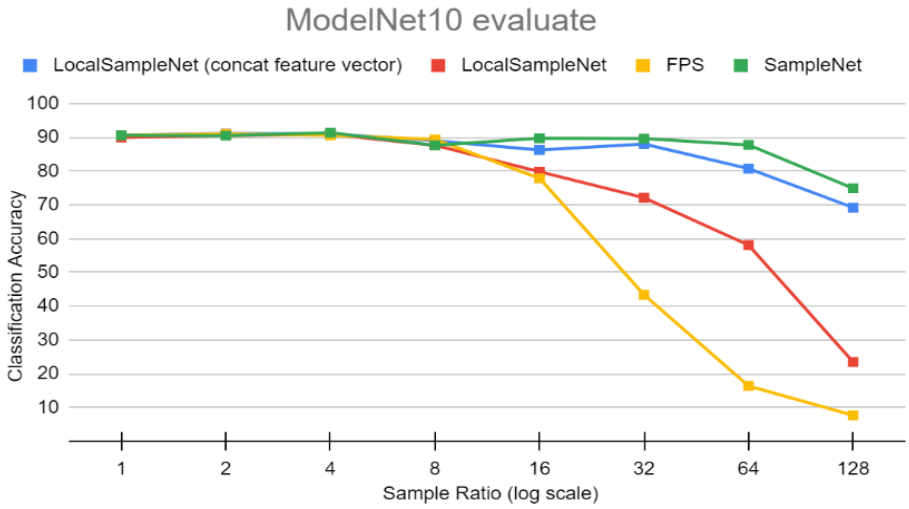


Figure 5 *MD10 evaluation for different sample method*

Compared to Samplenet, LocalSampleNet generalization is better, for example higher by 6.8% at sample ratio 32 on MD30 Figure 6.

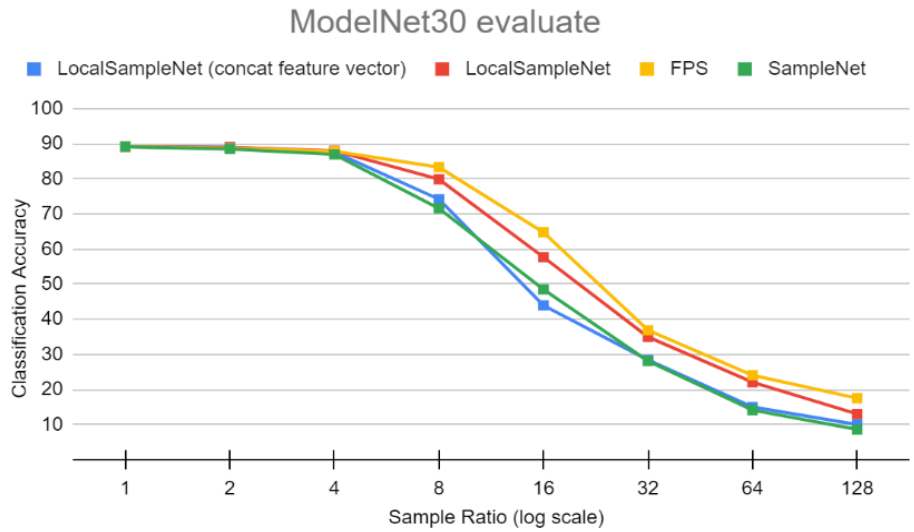


Figure 6 *MD30 evaluation for different sample method*

Overall, on MD40, SampleNet and LocalSampleNet **concat** act similar and are superior for an SR of above 32. For a lower value of SR, the FPS sampler and LocalSampleNet higher act similar and are superior. Figure 7.

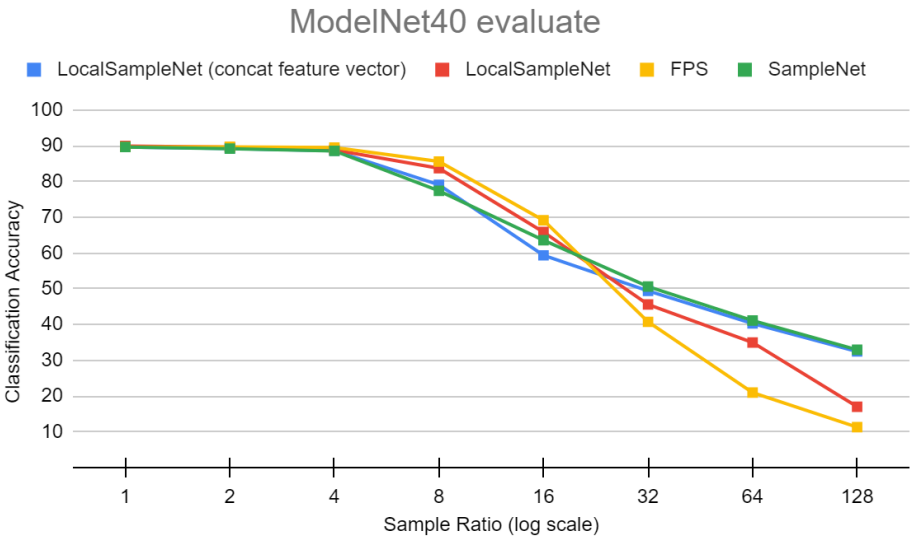


Figure 7 *MD40 evaluation for different sample metho*

4.3 Dropout Augmentation for Classification Task

Performance increase can be achieved with simple augmentation on the input data. While training the task classifier we notice that randomly reduced each point cloud by 0% to 87%, this is done by duplicated points, thus the input remains 1024 points. It turned out, that this augmentation enhanced the accuracy for all sampling method. We will sub name it **dropout task.** For example, on MD10 for sample ratio 32, *LocalSampleNet* (dropout task) accuracy is 86.9% and higher by 14.9% from non-dropout task.

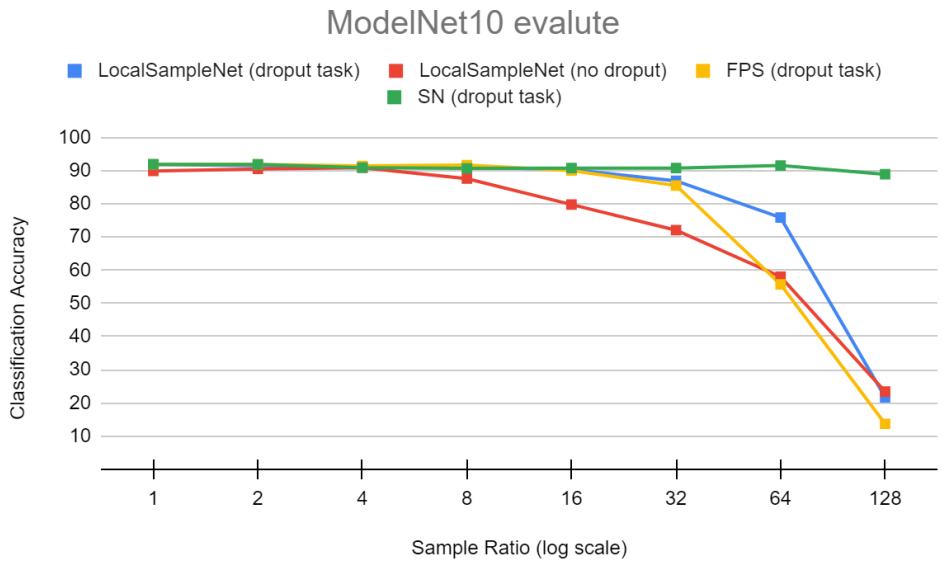


Figure 8 *MD10 evaluate dropout task with different simple method*

For MD30 the growth is even higher between dropout and non-dropout task. At sample ratio 32 LocalSampleNet with dropout increase by 33% and archived a 78.1% accuracy. That can be explained due to classifier training on MD40.

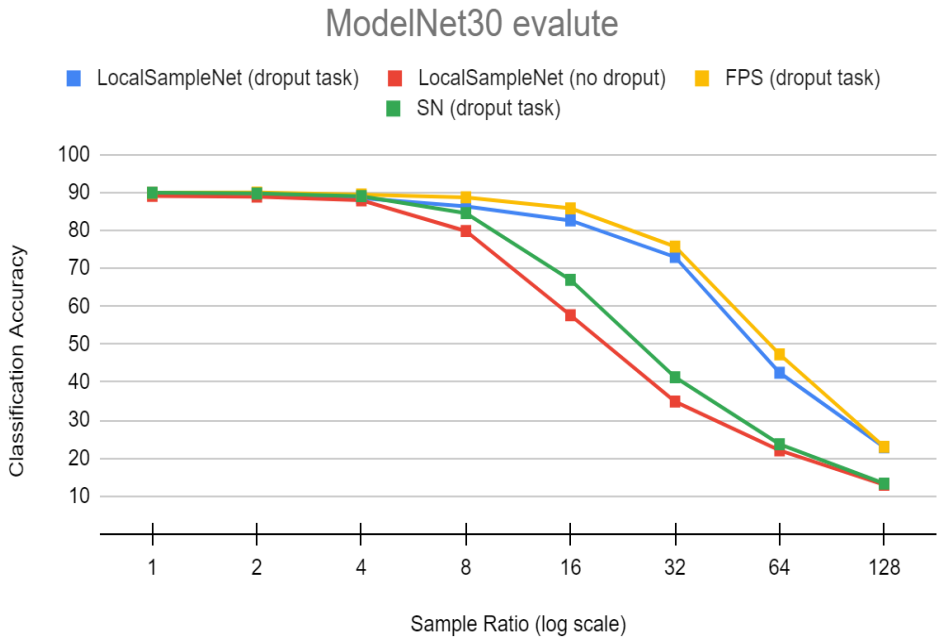


Figure 9 ***MD30 evaluate dropout task with different simple method***

Overall, For MD40 *LSN* with drop out, and the dropout task alone (with FPS sampling) achieve similar results. Compared to *LSN* vanilla and *SN*, the classification accuracy is higher.

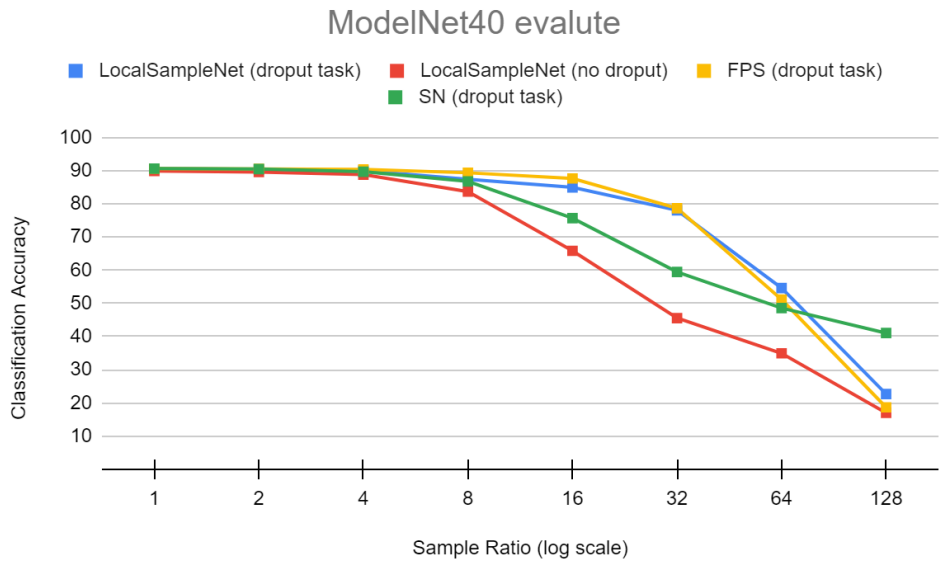


Figure 10 *MD40 evaluate dropout task with different sample methods*

4.4 SampleSeed

We examined a mini SampleNet layer which purpose is to choose seeds, instead of FPS Method, termed ***SampleSeed***, its MLP layers contains $[64,64,128,128]$ outputs size and Fully connected layers contain $[256,256,256,Sx3]$ .S symbols the amount of seeds to choose. We used partial loss from original SampleNet losses, given by:

$$\mathcal{L}_{SampleSeed} = \mathcal{L}_a(S,N)$$

Equation 7 *SampleSeed Loss*

Where S represent seeds, and N the original input point cloud. This loss ensures spread the seeds over the original point cloud.

Patch Number	Points per patch	Factor of SampleSeed loss	MD10	MD30	MD40
32	32	10	53.304	25.128	35.54
		1	**56.93**	**26.025**	**37.39**
		0.1	51.98	25.769	35.413
16	64	1	**62.88**	**25**	**38.938**
8	128	1	60.68	24.23	37.64
4	256	1	58.92	22.692	36.021

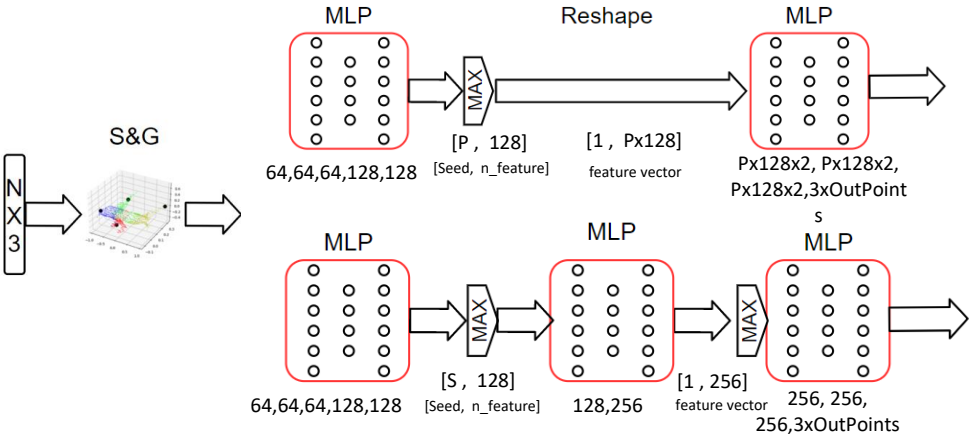
Table 3 *SampleSeed evaluation for sample ratio 32**.** Sample seeds loss element was added to the total loss with different factor. After that, it was tested with different patch number and points*

4.5 One feature vector – MLP method

We attempted to extract one feature vector by adding *MLP* layers, as can be seen in Figure 11, instead of reshaping the $P \times 128$ feature vector. As mentioned before, the *FPS* randomly selects the first point, therefor there is a random factor when trying to concatenate the feature vectors.

Feature vector extract	MD10	MD30	MD40
Reshape	**86.2**	**28.39**	**49.39**
MLP	85.57	25.13	47.204

Table 4 *Extraction one feature vector with two different methods.* ***Both trained and evaluated for sample ration 32, 8 patches and 128 points per patch.***



.

Figure 11. *One feature vector extract.* ***The top graph represents the original architecture* concatenate *feature vector. The bottom graph is MLP extract feature vector***

5 Conclusion

In this project we present LocalSampleNet, a novel local sampling neural network architecture.

We were able to improve the generalization ability compared to the global SampleNet, while maintaining high performances for structures that the network was trained on. Additionally, we noticed the importance and strength of using augmentations in the training pipeline. By using just, a dropout augmentation we were able to improve the local sampling abilities, without any further changes in the network architecture, compute power or inference data.

Future work can be focused on proceeding to improve the ability to extract local information, by using additional inputs such as the points normal or curvature.
Additional field can be point clouds registration. The registration process can benefit dramatically by incorporating local features information.

References

[1] Itai Lang, Asaf Manor, and Shai Avidan. "SampleNet: Differentiable Point Cloud Sampling." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7578-7588. 2020.

[2] Charles R. Qi, Li Yi, Hao Su, and Leonidas J Guibas. “Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In Advances in Neural Information Processing Systems, pp. 5099–5108,2017.

[3] C. Moenning and N. A. Dodgson. Fast Marching farthest point sampling. In Eurographics Poster Presentation, 2003. Technical Report 562, University of Cambridge Computer Laboratory.

[4] Kamousi, Pegah, Sylvain Lazard, Anil Maheshwari, and Stefanie Wuhrer. "Analysis of farthest point sampling for approximating geodesics in a graph." Computational Geometry 57 (2016): 1-7.

[5] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 652–660, 2017

[6] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to Sample. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2760–2769, 2019.

[7] Wu, Zhirong, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. "3d shapenets: A deep representation for volumetric shapes." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1912-1920. 2015.

[8] "GitHub pointnet_pointet2_pytorch by yanx27", https://github.com/yanx27/Pointnet_Pointnet2_pytorch

[9] "GitHub SampleNet by Itailang", https://github.com/itailang/SampleNet

[10] "GitHub PointNet2 by chalesq32", https://github.com/charlesq34/pointnet2