Equivariant vs. Invariant Layers: A Comparison of Backbone and Pooling for Point Cloud Classification

Abihith Kothapalli [1]   **Ashkan Shahbazi** [1]   **Xinran Liu** [1]   **Robert Sheng** [1]   **Soheil Kolouri** [1]

Editors: *S. Vadgama, E.J. Bekkers, A. Pouplin, S.O. Kaba, H. Lawrence, R. Walters, T. Emerson, H. Kvinge, J.M. Tomczak, S. Jegelka*

Abstract

Learning from set-structured data, such as point clouds, has gained significant attention from the machine learning community. Geometric deep learning provides a blueprint for designing effective set neural networks that preserve the permutation symmetry of set-structured data. Of our interest are permutation invariant networks, which are composed of a permutation equivariant backbone, permutation invariant global pooling, and regression/classification head. While existing literature has focused on improving equivariant backbones, the impact of the pooling layer is often overlooked. In this paper, we examine the interplay between permutation equivariant backbones and permutation invariant global pooling on three benchmark point cloud classification datasets. Our findings reveal that: 1) complex pooling methods, such as transport-based or attention-based poolings, can significantly boost the performance of simple backbones, but the benefits diminish for more complex backbones, 2) even complex backbones can benefit from pooling layers in low data scenarios, 3) surprisingly, the choice of pooling layers can have a more significant impact on the model's performance than adjusting the width and depth of the backbone, and 4) pairwise combination of pooling layers can significantly improve the performance of a fixed backbone. Our comprehensive study provides insights for practitioners to design better permutation invariant set neural networks. Our code is available at https://github.com/mint-vu/backbone_vs_pooling.

[1]Department of Computer Science, Vanderbilt University, Nashville, TN, USA. Correspondence to: Abihith Kothapalli <abi.kothapalli@vanderbilt.edu>.

1. Introduction

Set classification is a challenging problem in machine learning that has numerous real-world applications, including computer vision (Hu et al., 2012; Lu et al., 2015; Wang et al., 2015; Ye et al., 2020), natural language processing (Bahdanau et al., 2014; Liu & Lapata, 2018; Gong et al., 2020; Zhang et al., 2021), and bioinformatics (Skianis et al., 2020; Ren et al., 2021; Kim, 2022). Recently, there have been significant developments in 3D data acquisition, leading to increased interest in representation learning from point cloud data, which consists of sets of unordered 3D points. However, learning from point clouds presents inherent challenges, such as noise, occlusion, and irregularity, which can make representation learning difficult (Xiang et al., 2021; Qi et al., 2017a; Hua et al., 2018; Liu et al., 2019; Qi et al., 2017b; Prokudin et al., 2019; Achlioptas et al., 2018; Hassani & Haley, 2019; Ravanbakhsh et al., 2017).

Most existing methods for point cloud classification can be grouped into three categories: voxel-based, projection-based, and point-based methods. Voxel-based methods are volumetric approaches that rasterize point clouds onto a three-dimensional grid and utilize 3D convolutional neural networks (CNNs) to learn from the rasterized data (Ben-Shabat et al., 2018; Hermosilla et al., 2018; Maturana & Scherer, 2015; Hua et al., 2018; Mao et al., 2019; Zhang et al., 2019). Projection-based approaches project/render the point cloud onto 2D images (e.g., depth maps) from multiple orthogonal views and then process these images with 2D CNNs (Kanezaki et al., 2018; Yu et al., 2018; Su et al., 2015; Mohammadi et al., 2021; Goyal et al., 2021; Li et al., 2018b). Lastly, point-based methods involve permutation invariant neural networks based on the principles of geometric deep learning (Bronstein et al., 2021). These methods (Xiang et al., 2021; Wang et al., 2019; Qi et al., 2017a; Lee et al., 2019b; Li et al., 2018a; Shen et al., 2018; Dominguez et al., 2018; Liu et al., 2019) utilize permutation invariant functions to exploit the inherent permutation symmetry in point clouds. They often involve using a permutation equivariant (or semi-equivariant) backbone, followed
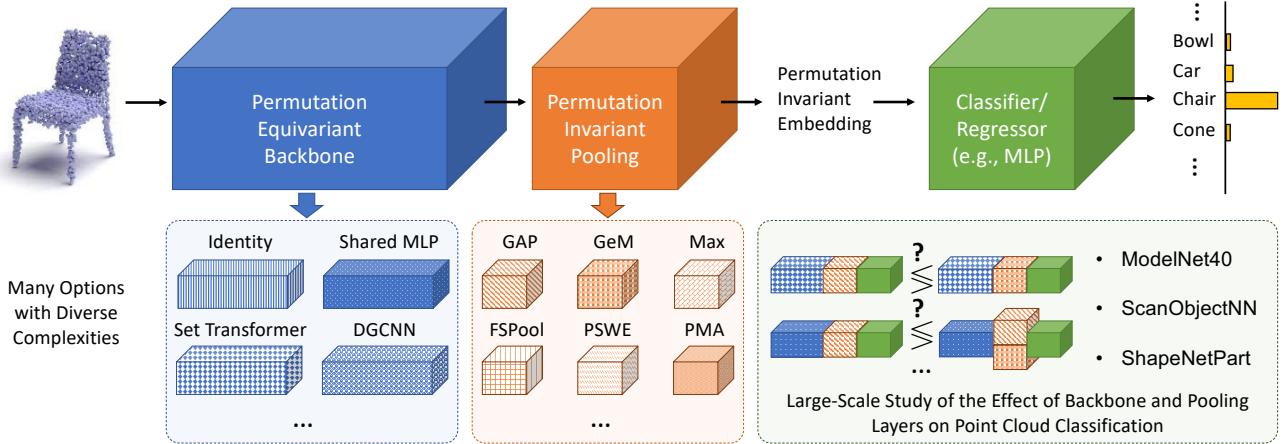
Figure 1: **Investigating the Impact of Backbone and Pooling Combinations on Point Cloud Data.** Through comprehensive experiments on three point cloud benchmarks, we evaluate the performance of models using different combinations of permutation equivariant backbones with permutation invariant pooling techniques. Additionally, we demonstrate the effectiveness of combining specific poolings to improve model performance. Our study provides insights into the benefits of equivariant and invariant layers for point cloud analysis.

by a permutation invariant global pooling to obtain a permutation invariant representation that is then passed into a classifier/regressor. This paper focuses on point-based approaches for point cloud classification rooted in geometric deep learning.

The majority of existing research in point cloud classification has focused on designing novel neural architectures, specifically permutation equivariant (or semi-equivariant) backbones (Wang et al., 2019; Qi et al., 2017a; Zaheer et al., 2017; Xiang et al., 2021; Lee et al., 2019b), while some recent work has highlighted the significance of pooling layers (Murphy et al., 2018; Zhang et al., 2020; Naderializadeh et al., 2021; Wang et al., 2022; Lee et al., 2019b; Wang et al., 2020b; Skianis et al., 2020; Mialon et al., 2021). Our objective in this paper is to investigate the intricate interplay between backbone architecture and pooling approach on model performance in point cloud classification. We conduct extensive studies on three prominent point cloud classification benchmark datasets — ModelNet40 (Wu et al., 2015), ScanObjectNN (Uy et al., 2019), and ShapeNetPart (Chang et al., 2015) — to explore the impact of model architectural choices and complexities on performance.

In our experiments, we aimed to investigate the interplay between the choice of permutation equivariant backbones and permutation invariant pooling approaches on point cloud classification performance. To this end, we evaluated seven different permutation equivariant backbones: Identity; DeepSets (Zaheer et al., 2017); (Induced) Set Attention Blocks (SAB and ISAB) (Lee et al., 2019b); Dynamic Graph CNN (DGCNN) (Wang et al., 2019); PointNet (Qi et al., 2017a); and CurveNet (Xiang et al., 2021). For poolings, we consider eleven different permutation invariant pooling approaches: Global Average Pool (GAP); Generalized

Mean (GeM) (Radenović et al., 2018); Max and k-Max (Kim, 2014); Pooling by Multihead Attention (PMA) (Lee et al., 2019b); Global Multi-Head Attentive (GMHA) and Multi-Resolution Multi-Head Attentive (MMHA) poolings (Wang et al., 2020b); Pooling by Sliced-Wasserstein Embedding (PSWE) (Naderializadeh et al., 2021) (both learnable and non-learnable versions); Featurewise Sort Pooling (FSPool) (Zhang et al., 2020); and RepSet (Skianis et al., 2020). While there have been many recent advancements in point cloud classification and segmentation models beyond what we discuss here, the purpose of this work is to focus only on permutation equivariant backbones and permutation invariant pooling layers, and hence many of these recent advancements fall outside the scope of this work. To ensure fair comparisons and avoid the effects of auxiliary factors like evaluation schemes, data augmentation strategies, and loss functions, we devised a unified training and evaluation scheme. We evaluated the performance of all pairs of backbones and pooling methods (a total of 77 models) on three benchmark datasets, namely ModelNet40 (Wu et al., 2015), ScanObjectNN (Uy et al., 2019), and ShapeNetPart (Chang et al., 2015). We also investigated the effect of using multiple pooling layers for fixed backbones and provided the performance of the 77 models under full and restricted training data.

Our findings can be summarized as follows. First, we observed a performance gap between traditional permutation invariant pooling approaches and more recent OT-based and attention-based pooling approaches. This gap is most prominent with simpler permutation equivariant backbones and becomes less significant as the backbone's complexity increases. Additionally, we demonstrated that this performance gap is more pronounced when working with limited

amounts of data. Second, we discovered that OT-based methods exhibit less sensitivity to the sample size, providing stable performance even in low-data scenarios, whereas attention-based methods exhibit more variation in their performance. Third, we observed that pooling layers play a more critical role in performance of DeepSets (Zaheer et al., 2017) and Set Transformers (Lee et al., 2019b) compared to the width and depth of the backbone. Lastly, we observed that certain permutation invariant pooling layers are complementary, and combining them can significantly enhance the model's performance.

Contributions. We demonstrate that: a) Permutation invariant pooling layers play a crucial role in point cloud classification. b) Transport-based pooling layers exhibit less sensitivity to training data size. Models with simple and shallow permutation equivariant backbones, combined with advanced pooling techniques, perform well, particularly in low-data scenarios. c) In the context of DeepSets Zaheer et al. (2017) and Set Transformers (Lee et al., 2019b), the choice of the pooling layer has a greater impact on performance than adjustments in backbone depth and width. d) Pairing specific pooling layers can result in significant performance improvements. Overall, our study provides valuable insights for the community in designing effective permutation invariant models and raises further awareness on the importance of global pooling layers.

2. Related Work

Numerous recent studies have been dedicated to addressing point set and graph classification problems by devising neural architectures that exhibit permutation invariance. These architectures typically consist of multiple permutation equivariant layers coupled with permutation invariant pooling operations. In the context of our focus on point cloud classification, here we provide a concise summary of recent advancements in developing permutation equivariant backbones and invariant pooling techniques. We also note that this work aims to focus only on permutation equivariant backbones and permutation invariant pooling layers; in this regard, while there have been many recent advancements in point cloud classification and segmentation models, even outperforming some of the methods we consider here, these models are not always permutation equivariant, and hence we do not consider them in this work.

Permutation Equivariant Backbones. In point clouds, the input features for each point typically consist of coordinates, occasionally supplemented by a surface normal. However, these input features alone often lack sufficient descriptions of shape geometries, rendering them insufficient for classification purposes. To address this limitation, permutation equivariant backbones play a crucial role by enabling the aggregation and enrichment of these features for each point. This process results in enhanced features

encompassing local and global information, providing a more comprehensive shape description. PointNet (Qi et al., 2017a) and its subsequent extension, PointNet++ (Qi et al., 2017b), have emerged as pioneering network architectures for point cloud classification. PointNet utilizes permutation equivariant blocks, which consist of a spatial/feature transformer network (Jaderberg et al., 2015), followed by a shared multi-layer perceptron (MLP) and featurewise max-pooling operations. These computational blocks in PointNet closely resemble the backbone of the DeepSets architecture (Zaheer et al., 2017), wherein a shared MLP is applied to the set elements to generate a permutation equivariant backbone.

Shared MLP-based permutation equivariant backbones, while widely used, have inherent limitations in their representation capabilities (Lee et al., 2019b), as they lack the ability to facilitate interactions or message passing between neighboring points. Conversely, alternative methods such as PointCNN (Li et al., 2018b), Set Transformers (Lee et al., 2019b), Dynamic Graph CNN (DGCNN) (Wang et al., 2019), and CurveNet (Xiang et al., 2021) offer more flexible permutation equivariant backbones by enabling message passing between set elements. In particular, the Set Transformer framework, examples of which include the Set Attention Block (SAB) and Induced Set Attention Block (ISAB) methods (Lee et al., 2019b), leverages the powerful self-attention mechanism (Vaswani et al., 2017) and its variations as the computational building blocks for the backbone. Similarly, DGCNN (Wang et al., 2019) modifies the message passing between input points by dynamically constructing a graph in each layer and utilizing edge convolutions. Lastly, CurveNet (Xiang et al., 2021) employs a guided walk on the point cloud to identify curve groupings, enabling more informative message passing within the network architecture.

Permutation Invariant Poolings. Permutation invariant pooling layers play a pivotal role in geometric deep learning, where the sum, average, and max/min functions are widely acknowledged as the simplest and commonly used permutation invariant functions in the literature. However, recent advancements have introduced more sophisticated permutation invariant pooling layers that offer enhanced performance. This emerging work stems from the observation that traditional pooling layers, such as Global Average Pooling (GAP), may not adequately capture the feature distribution extracted by the permutation equivariant backbone. To address this, researchers have proposed novel approaches such as covariance pooling, introduced by Acharya et al. (2018) and Wang et al. (2020a), which aims to capture the second moment of the feature distribution. Similarly, Radenović et al. (2018) proposed Generalized Mean (GeM) pooling, which approximates higher moments of the feature distribution while mitigating the computational cost associated

with covariance pooling. Murphy et al. (2018) introduced Janossy pooling, which represents permutation invariant functions as the average of "permutation-sensitive" functions applied to all reorderings of the input sequence. Furthermore, recent studies have explored permutation invariant pooling layers based on optimal transport (OT) theory, extending beyond the second moment of distributions. For example, Kolouri et al. (2021) and Mialon et al. (2021) concurrently introduced permutation invariant pooling layers for graph neural networks using Wasserstein embedding (also known as linearized OT (Wang et al., 2013; Moosmüller & Cloninger, 2023)). Naderializadeh et al. (2021) expanded upon the Wasserstein embedding framework with their proposal of Pooling by Sliced Wasserstein Embedding (PSWE), a suitable permutation invariant function for end-to-end learning that encodes the backbone's feature distribution. Notably, Zhang et al. (2020) had previously introduced Featurewise Sort Pooling (FSPool), which can be considered a specific case of PSWE. Skianis et al. (2020) developed RepSet and its approximation version, ApprRepSet, which leverages bipartite matching and is similar to OT-based pooling layers.

Attention-based permutation invariant pooling layers have emerged as another family of pooling layers and have gained significant attention in geometric deep learning due to their ability to capture complex relationships and dependencies among features while maintaining permutation invariance. These pooling layers leverage the attention mechanisms (Vaswani et al., 2017), allowing the model to assign weights to different features based on their relevance dynamically. The attention mechanism enables the pooling layer to aggregate important features while suppressing less significant ones, thus enhancing the overall representation power of the layer. Several studies have proposed attention-based permutation invariant pooling layers for geometric deep learning tasks. For example, Lee et al. (2019a) presented Self-Attention Graph Pooling, where attention coefficients are learned to guide the pooling process based on node-level features. Similarly, Wang et al. (2022) introduced graph multi-head pooling which utilizes attention as its core pooling mechanism. Lee et al. (2019b) introduced pooling by multi-head attention (PMA) and Wang et al. (2020b) proposed global multi-head attentive (GMHA) and multi-resolution multi-head attentive (MMHA) poolings. These attention-based pooling layers not only improve the model's ability to capture important features but also provide a mechanism to adaptively aggregate information from different nodes or regions in a permutation invariant manner.

Although significant progress has been made in developing novel permutation equivariant backbones and permutation invariant pooling layers in geometric deep learning, the interaction between these two modules has not been thoroughly investigated in prior works. In our paper, we aim to bridge this gap by conducting an extensive study on the interplay between the backbone and pooling layers on three benchmark point cloud classification datasets. By analyzing the performance of different combinations of backbones and pooling layers, we provide valuable insights into designing effective and efficient geometric deep learning models. Our study aims to shed light on the role of these modules and their interaction in point cloud classification tasks.

3. Experiments

Our study encompasses a comprehensive set of experiments aimed at addressing key questions in the context of point cloud classification. Firstly, we investigated the impact of combining various permutation invariant pooling techniques with permutation equivariant backbone architectures on classification performance. Secondly, we explored how this combination affects learning when dealing with limited training data. Thirdly, we analyzed the effects of adjusting the depth and width of a backbone, specifically DeepSets and Set Transformer (in particular, the SAB method), on the network's performance when different pooling layers are utilized. Lastly, we examined the benefits of pairing specific pooling layers with a given backbone architecture.

We conducted experiments on three point cloud classification benchmarks: ModelNet40 (Wu et al., 2015), ScanObjectNN (Uy et al., 2019), and ShapeNetPart, which is the point cloud variant of the ShapeNet dataset (Chang et al., 2015). To avoid nuisance training variations, like different types of augmentations, different loss functions, training hyperparameters, etc., we designed a unified experimental setup for all datasets and for all models. In what follows, we first provide a brief description of the datasets, then review the permutation equivariant backbones followed by the different pooling layers and the classifier head we used in this work.

3.1. Datasets

ModelNet40 (Wu et al., 2015) is a popular benchmark in 3D object recognition, comprising 12,311 CAD models from 40 object categories, with around 300 instances per category. This dataset offers diverse objects in terms of shape and size, making it well-suited for evaluating the performance of 3D point cloud classification models. Each instance in the dataset consists of 2048 points.

ScanObjectNN (Uy et al., 2019) is a real-world point cloud object dataset derived from scanned indoor scenes. It encompasses nearly 15,000 objects across 15 categories, with 2902 unique object instances. In our experiments, we focus solely on utilizing the global coordinates of each point per sample, disregarding other attributes such as normals, color attributes, semantic labels, and part annotations provided by the original ScanObjectNN dataset. Like ModelNet40, each sample in ScanObjectNN comprises 2048 points.

ShapeNetPart is the point cloud counterpart of the ShapeNet (Chang et al., 2015) dataset, comprising more than 15,000 3D models across 16 object categories, including furniture, cars, airplanes, and animals. The objects are originally represented as 3D meshes, but here we consider the mesh nodes as a 3D point cloud, disregarding the edges. Each model in this dataset contains a variable number of points, typically ranging from approximately 500 to 3000 points for a given object.

For all datasets, we randomly sampled 1024 points for each object to construct a batch. For the ModelNet40 and ScanObjectNN datasets, we sampled these points without replacement, and for the ShapeNetPart dataset, we sampled these points with replacement (as there may be fewer than 1024 points initially). Furthermore, we followed the work of Naderializadeh et al. (2021) and applied random translation, rotation, and jittering for data augmentation in all experiments.

3.2. Model Blueprint

Permutation Equivariant Backbones. In our experiments, we carefully selected diverse backbones with varying forward time, backward time, and model size. To establish a baseline, we utilized the Identity backbone (i.e., lack of a backbone) with different pooling techniques. Subsequently, we conducted similar experiments on several other backbones, including MLP (as used in DeepSets), SAB (Lee et al., 2019b), ISAB (Lee et al., 2019b), DGCNN (Wang et al., 2019), PointNet (Qi et al., 2017a), and CurveNet (Xiang et al., 2021). For detailed information regarding the hyperparameters of the backbones, we refer readers to Appendix B.1.

Permutation Invariant Global Pooling. We explored the role of different common and state-of-the-art pooling methods in learning from sets. Our experiments include the classic pooling techniques, including GAP, Max, k-Max (Kim, 2014), and GeM (Radenović et al., 2018), as well as more complex pooling methods like Learnable and Frozen (Non-Learnable) versions of PSWE (Naderializadeh et al., 2021) (which we denote L-PSWE and F-PSWE, respectively), FSPool (Zhang et al., 2020), and ApproxRepSet (Skianis et al., 2020). We also considered attention-based pooling mechanisms, such as PMA (Lee et al., 2019b), GMHA (Wang et al., 2020b), and MMHA (Wang et al., 2020b), on different backbones. Further details on pooling hyperparameters can be found in Appendix B.2.

Classifier Head. For all models, our classifier consists of a feed-forward network with three hidden layers, each consisting of 128 nodes, and a final layer with its width equal to the number of classes in the dataset. We apply batch normalization and leaky ReLU activation to each hidden layer, as well as dropout after each hidden layer.

3.3. Training and Evaluation Settings

In our experiments, we allocated 1% of the training data as the validation set. For each model, we simultaneously trained the backbone, pooling, and classifier parameters using the Adam optimizer (Diederik & Jimmy, 2014) with cross entropy loss (Ackley et al., 1985) and batches of 32 samples. The initial learning rate was set to 8×10^{-4}, and we applied a learning rate decay of $\gamma = 0.5$ every 50 epochs for all experiments. Early stopping was considered when the validation loss did not improve for 20 consecutive epochs, and training was capped at 500 epochs if early stopping was not triggered. We used classification accuracy on the test set as our evaluation criterion. Each experiment was conducted with different random seeds three times, and we report the average performance per model. We utilized GPU clusters with 16 NVIDIA RTX A6000 GPUs, AMD EPYC 7713 64-Core Processor, and 256GB of memory.

4. Results

	Models	Id	MLP	SAB	ISAB	DGCNN	PN	CN	P Avg.
ModelNet40	GAP	4.01	40.02	83.81	83.35	85.80	84.67	89.93	67.37
	GeM	26.52	38.74	85.74	83.96	89.76	85.86	**90.37**	71.56
	Max	28.46	31.77	**85.86**	85.80	89.72	**86.93**	88.82	71.05
	k-Max	27.76	30.25	84.70	83.48	89.94	**87.26**	90.18	70.51
	PMA	35.20	48.96	75.96	63.42	87.99	85.84	88.78	69.45
	F-PSWE	**80.57**	80.25	83.97	**85.46**	90.65	**86.98**	89.50	**85.34**
	L-PSWE	**78.61**	**81.65**	82.53	83.19	90.40	86.39	**90.79**	84.79
	FSPool	39.93	41.02	**86.22**	83.57	90.64	84.88	89.54	73.69
	GMHA	**78.50**	75.12	79.93	**85.62**	90.38	85.32	90.34	83.60
	MMHA	78.31	**78.33**	85.82	84.79	89.41	86.62	**90.85**	84.87
	ApprRepSet	29.83	39.62	84.46	84.99	89.53	84.95	89.97	71.91
	B Avg.	46.16	53.25	83.54	82.51	89.48	85.97	89.92	
ScanObjectNN	GAP	13.39	33.34	66.17	60.78	76.41	52.71	73.18	53.71
	GeM	38.12	37.02	65.11	62.64	73.56	60.44	73.83	58.67
	Max	30.67	36.68	66.38	66.04	77.88	**62.33**	77.64	59.66
	k-Max	31.32	33.96	65.49	**64.32**	73.70	61.74	**77.47**	58.29
	PMA	38.26	34.92	53.43	42.58	73.21	55.32	**78.50**	53.75
	F-PSWE	**51.92**	**56.35**	65.28	63.19	76.65	**63.08**	76.99	**64.78**
	L-PSWE	**50.27**	**55.36**	66.21	62.71	76.24	51.99	75.86	62.66
	FSPool	38.77	40.08	66.07	**63.56**	73.18	55.43	74.38	58.78
	GMHA	**48.94**	**52.13**	66.69	60.75	**76.99**	63.19	76.10	63.54
	MMHA	45.95	47.84	62.81	60.85	70.05	61.50	77.27	60.90
	ApprRepSet	37.19	33.83	**66.96**	61.54	68.65	60.30	75.69	57.74
	B Avg.	38.62	41.95	64.60	60.81	74.23	58.91	76.08	
ShapeNetPart	GAP	29.11	88.33	97.35	86.09	97.81	**97.85**	98.26	84.97
	GeM	70.21	87.59	97.14	94.44	98.37	97.35	98.19	91.90
	Max	66.34	79.59	97.14	**97.39**	98.37	**97.91**	97.53	90.61
	k-Max	65.92	80.96	97.62	96.76	98.37	97.25	**98.44**	90.76
	PMA	64.66	93.52	97.36	**97.83**	98.06	97.01	98.33	92.39
	F-PSWE	**96.35**	**97.25**	98.05	94.09	98.44	97.46	**98.47**	97.16
	L-PSWE	**96.49**	96.13	**97.79**	95.75	**98.51**	97.53	98.02	**97.17**
	FSPool	83.22	92.58	97.60	**96.80**	98.40	**97.74**	98.26	94.94
	GMHA	**95.16**	94.95	97.53	96.52	98.01	97.42	**98.47**	96.87
	MMHA	94.54	**95.79**	98.15	96.52	**98.61**	97.18	98.40	97.03
	ApprRepSet	82.23	85.37	97.35	96.23	**98.58**	97.47	98.37	93.66
	B Avg.	76.75	90.19	97.55	95.31	**98.32**	97.47	98.25	

Table 1: We present the results of 77 models across three datasets, highlighting the top three performers for each backbone and the best-performing pooling layer and backbone on average. Here, "Id" refers to identity, "MLP" refers to the shared MLP backbone in DeepSets, "PN" refers to PointNet, and "CN" refers to CurveNet.
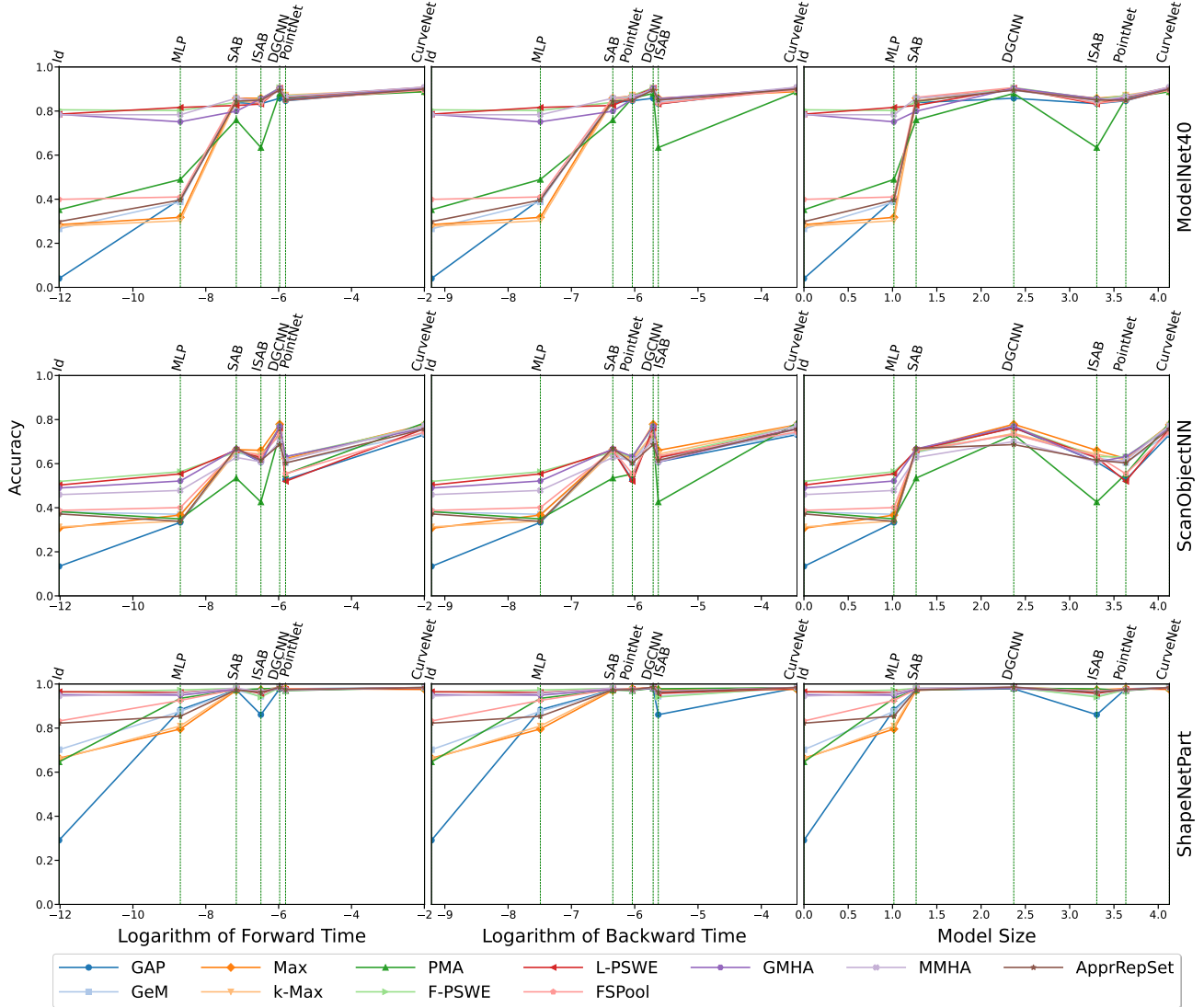
Figure 2: **Backbone vs. Pooling.** This figure provides a visualization of the models' performances, reported in Table 1, as a function of the backbone complexity. We employed three indicators as proxies for backbone complexity: average forward time, average backward time, and model size. Each row represents the results from one of the datasets.

Pooling vs. Backbone for Point Cloud Classification.
We conducted experiments on the pairwise combination of seven permutation equivariant backbones and eleven permutation invariant pooling layers, as described in Section 3. Each model was trained on the three benchmark datasets, and we repeated each experiment three times, resulting in 231 trained models per dataset. The average test accuracy of the models is reported in Table 1. Additionally, we provide the average performance of each backbone across all pooling techniques and the average performance of each pooling technique across all backbones. The top three pooling layers for each backbone and the best overall backbone and pooling methods are shown in bold.

In our experiments, where we controlled for all other parameters, the impact of pooling layers on performance be-

came evident. The OT-based pooling methods, specifically F-PSWE, consistently achieved excellent results across different backbones. The attention-based pooling layers also demonstrated impressive performance. Intriguingly, we observed that both the OT-based and attention-based pooling techniques yielded satisfactory performance even in the absence of a backbone (i.e., with an identity backbone), highlighting the modeling flexibility of these methods.

To further analyze the findings presented in Table 1, we examined the performance of pooling layers in relation to the complexity of the backbones. We utilized three indicators as proxies for backbone complexity: average forward time, average backward time, and model size. Figure 2 displays the performance of different pooling layers for each dataset as a function of the backbones' complexities. Notably, we
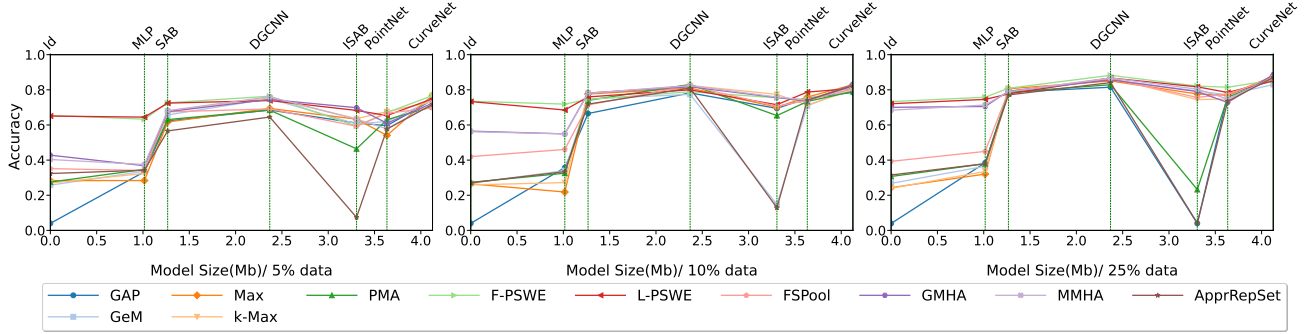
Figure 3: **Backbone vs. Pooling under Limited Training Data.** The models' performances on ModelNet40, when using 5% (left), 10% (middle), and 25% (right) of the training data. Notably, the OT-based methods show less sensitivity to the sample size.

continue to observe a substantial improvement when utilizing OT-based pooling layers, such as L-PSWE and F-PSWE, and attention-based pooling layers like GMHA and MMHA, in conjunction with the backbones of lower complexities. However, as the complexity of the backbone increases, the advantage conferred by these pooling layers diminishes.

Learning with Limited Training Data. In our subsequent analysis, we investigated the interplay between pooling methods and backbones when training with limited data. We randomly sampled 5%, 10%, and 25% of the training data from the ModelNet40 dataset and conducted our experiments accordingly. Figure 3 depicts the performance of the models as a function of model size. Notably, we observed the robustness of OT-based pooling methods under limited data conditions. Specifically, for the MLP backbone, L-PSWE and F-PSWE displayed less sensitivity to the training set size, whereas GMHA and MMHA exhibited higher sensitivity. Another significant finding was that while the impact of pooling layers diminished as backbone complexity increased when training on the full dataset, these differences became more pronounced when training with limited data, even for complex backbones.

Depth vs. Width. For models such as MLP and SAB, the width and depth of the model provide control over its size and complexity. It is natural to inquire about the impact of the pooling layer in relation to the backbone size, given a fixed architecture type. To address this, we constructed models with various depths (ranging from 1 to 5) and widths (64, 128, 512, and 1024) for a fixed architecture type. This resulted in 25 different backbones, which, combined with the 11 pooling layers and three repetitions, yielded a total of 825 models per architecture type. We conducted this experiment on the ModelNet40 dataset and presented the results for MLP in Figure 4. Once again, we observe the overall superiority of OT-based pooling methods, L-PSWE and F-PSWE, with attention-based approaches such as GMHA and MMHA following closely behind. Similar results for SAB are provided in Appendix C.

Another notable observation is that while adjusting the width and depth of the backbone impacts the performance to some extent, the most significant performance improvement is achieved through the utilization of better permutation invariant pooling techniques, particularly for DeepSets and Set Transformers.
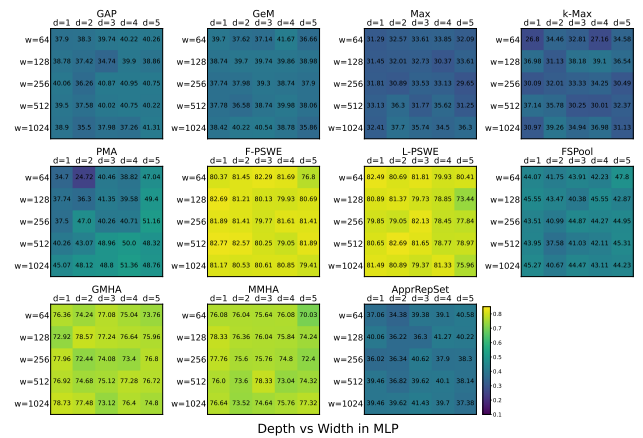

Figure 4: **Depth vs. Width.** The performance of each pooling method on MLP backbones with varied widths and depths is illustrated in the figure. The results consistently highlight the superior performance of OT-based poolings, particularly F-PSWE and L-PSWE, across different backbone sizes. Additionally, attention-based poolings GMHA and MMHA exhibit competitive performance, closely trailing behind the OT-based methods.

Pairing Pooling Layers. Combining different global pooling methods, such as GAP and Max pooling, for point cloud classification is a common approach used across a variety of models (Wang et al., 2019; Xiang et al., 2021; Qi et al., 2017a;b; Goyal et al., 2021; Su et al., 2015). This is done by directly concatenating the output of each pooling method prior to the classification head. Here, we seek to understand which pairings of the pooling layers are complementary.

We evaluated pairs of pooling layers for fixed backbones

on the three datasets, resulting in 55 models per backbone and per dataset. Specifically, we used MLP and SAB as our backbones. Figure 5 displays the performance of models with different pairs of pooling layers for the MLP and the SAB backbones on the ModelNet40 dataset. Additionally, we provide the performance of each individual pooling layer in the bottom row of each plot for the sake of comparison. As expected, we can see that not all pairings are complementary, while many are. In particular, the performance of the F-PSWE pooling method with MLP backbone, 75.16%, can be boosted to 80.29% when complemented with MMHA. We also repeated this experiment on the ScanObjectNN and ShapeNet datasets, and the results can be found in Appendix D.
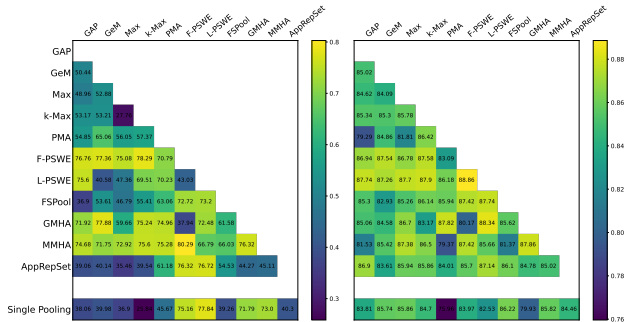
Figure 5: **Effect of Paired Pooling Methods.** Test set classification accuracy for models trained with the same backbone and different pairs of pooling layers. Results are shown for MLP (left) and SAB (right) backbones. The bottom rows in both plots represent the performance of the same model with a single pooling layer.

5. Discussions and Concluding Remarks

Point-based methods for deep point cloud classification rely on permutation invariant neural networks, which typically consist of a permutation equivariant backbone, global permutation invariant pooling, and a shallow classifier. Recent models have achieved improved accuracies on benchmark datasets. However, it has been noted that the observed progress may not solely stem from better network architecture designs but also from various auxiliary factors such as evaluation schemes, data augmentation strategies, and loss functions, making it challenging to attribute improvements solely to the network architecture itself (Goyal et al., 2021). Our work explores the intricate interplay between backbone architecture and pooling approaches in point cloud classification to achieve better network architectures. We aim to understand how these factors impact model performance in a controlled setting where training and evaluation conditions are standardized across all models. This allows us to disentangle the effects of network architecture from other factors and gain deeper insights into the relationship between backbone architecture, pooling methods, and classification

performance in point cloud analysis.

We trained more than 4,000 models on the three benchmark datasets – ModelNet40, ScanObjectNN, and ShapeNetPart – to answer the following questions: How important is the effect of the backbone versus the pooling layer in point cloud classification? How robust are different models when learning with limited data? What is the effect of width and depth of the permutation equivariant backbones for different pooling layers in DeepSets and Set Transformers? And lastly, would pairing permutation invariant pooling layers affect the models' performances? Our findings highlight the performance gap between traditional and advanced pooling methods, the robustness of models under limited data, and the critical role of pooling layers in DeepSets and Set Transformers. Further, we observed that pairing pooling layers could lead to enhanced model performance.

Our results consistently showed that OT-based pooling methods, such as F-PSWE and L-PSWE, outperform traditional pooling techniques across various backbones and datasets. The superior performance of OT-based methods can likely be attributed to their ability to capture complex relationships and distributions within the feature space, providing a richer and more informative representation of the point cloud data. This suggests that for applications requiring high accuracy and robustness, incorporating OT-based pooling layers is highly beneficial. Meanwhile, attention-based pooling methods, including GMHA and MMHA, also demonstrated strong performance. These methods dynamically assign weights to different features based on their relevance, effectively focusing on the most informative parts of the point cloud. This dynamic weighting mechanism allows attention-based pooling to adaptively aggregate important features, which is particularly useful in scenarios with complex or noisy data.

We also observed that the performance benefits of complex pooling methods are more pronounced with simpler backbones. For instance, when using only Identity as our backbone, advanced pooling methods like F-PSWE and GMHA provide significant performance improvements. However, as the complexity of the backbone increases, the relative gains from these sophisticated pooling methods diminish. This indicates that while complex backbones can learn rich features independently, simpler backbones can substantially benefit from advanced pooling techniques to enhance their representational power.

Our work offers many potential insights for practitioners tackling point cloud classification tasks. First, we find that combining different pooling methods can yield significant performance improvements. This complementary effect arises because different pooling methods capture different aspects of the feature distribution, leading to a more comprehensive aggregation of the point cloud data. Therefore, for

practitioners, experimenting with combinations of pooling layers may yield worthwhile performance improvements. In scenarios with limited training data, OT-based pooling methods exhibit less sensitivity to the sample size compared to alternative pooling methods. This robustness makes OT-based pooling particularly suitable for applications where data is scarce or expensive to collect. Finally, we observe that while both the complexity of the backbone and the pooling layers contribute to overall model performance, it is crucial to strike a balance between these components. For simpler backbones, investing in more sophisticated pooling methods can lead to substantial gains. Conversely, for complex backbones, simpler pooling methods may suffice, allowing computational resources to be allocated elsewhere in the model.

Ultimately, point cloud classification remains a significant area of interest in the machine learning community, with continuous advancements being made. Several works which we did not originally consider in our experiments, such as (Wu et al., 2022; Ma et al., 2022; Chen et al., 2023; Qi et al., 2024), have presented novel state-of-the-art point cloud classification approaches that further enhance classification accuracy and robustness. Integrating these newer architectures into our experimental framework is thus an important direction for future work. Moreover, while the focus of this work is on permutation invariant model architectures, we additionally note the $SO(3)$ symmetry inherent to point cloud data. This symmetry motivates the use of methods that are invariant to rotation as well. While the use of rotation augmentation in our experimental setup aids in partially addressing this symmetry, recent works such as (Chen & Cong, 2022) and (Wang et al., 2024) offer promising results by explicitly incorporating rotation invariance into their model architectures. The integration of such methods into our work to achieve a more holistic understanding of point cloud classification approaches presents yet another valuable avenue for future research.

Acknowledgements

References

Acharya, D., Huang, Z., Pani Paudel, D., and Van Gool, L. Covariance pooling for facial expression recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 367–374, 2018.

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. Learning representations and generative models for 3D point clouds. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 40–49. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/achlioptas18a.html.

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. A learning algorithm for boltzmann machines. *Cognitive science*, 9 (1):147–169, 1985.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Ben-Shabat, Y., Lindenbaum, M., and Fischer, A. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152, 2018. doi: 10.1109/LRA.2018.2850061.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. Shapenet: An information-rich 3d model repository, 2015.

Chen, G., Wang, M., Yang, Y., Yu, K., Yuan, L., and Yue, Y. PointGPT: Auto-regressively generative pre-training from point clouds. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=rqE0fEQDqs.

Chen, R. and Cong, Y. The devil is in the pose: Ambiguity-free 3d rotation-invariant learning via pose-aware convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7472–7481, June 2022.

Diederik, P. and Jimmy, B. Adam: A method for stochastic optimization. iclr. *arXiv preprint arXiv:1412.6980*, 2014.

Dominguez, M., Dhamdhere, R., Petkar, A., Jain, S., Sah, S., and Ptucha, R. General-purpose deep point cloud feature extractor. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1972–1981. IEEE, 2018.

Gong, J., Teng, Z., Teng, Q., Zhang, H., Du, L., Chen, S., Bhuiyan, M. Z. A., Li, J., Liu, M., and Ma, H. Hierarchical graph transformer-based deep learning model for large-scale multi-label text classification. *IEEE Access*, 8:30885–30896, 2020.

Goyal, A., Law, H., Liu, B., Newell, A., and Deng, J. Revisiting point cloud shape classification with a simple and effective baseline. *International Conference on Machine Learning*, 2021.

Hassani, K. and Haley, M. Unsupervised multi-task feature learning on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Hermosilla, P., Ritschel, T., Vázquez, P.-P., Vinacua, À., and Ropinski, T. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018.

Hu, Y., Mian, A. S., and Owens, R. Face recognition using sparse approximated nearest points between image sets. *IEEE transactions on pattern analysis and machine intelligence*, 34(10):1992–2004, 2012.

Hua, B.-S., Tran, M.-K., and Yeung, S.-K. Pointwise convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 984–993, 2018.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Kanezaki, A., Matsushita, Y., and Nishida, Y. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5010–5019, 2018.

Kim, M. Differentiable expectation-maximization for set representation learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=MXdFBmHT4C.

Kim, Y. Convolutional neural networks for sentence classification, 2014.

Kolouri, S., Naderializadeh, N., Rohde, G. K., and Hoffmann, H. Wasserstein embedding for graph learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AAes_3W-2z.

Lee, J., Lee, I., and Kang, J. Self-attention graph pooling. In *International conference on machine learning*, pp. 3734–3743. PMLR, 2019a.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3744–3753, 2019b.

Li, J., Chen, B. M., and Lee, G. H. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9397–9406, 2018a.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. Pointcnn: Convolution on x-transformed points. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/f5f8590cd58a54e94377e6ae2eded4d9-Paper.pdf.

Liu, X., Han, Z., Liu, Y.-S., and Zwicker, M. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 8778–8785, 2019.

Liu, Y. and Lapata, M. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75, 2018.

Lu, J., Wang, G., Deng, W., Moulin, P., and Zhou, J. Multimanifold deep metric learning for image set classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1137–1145, 2015.

Ma, X., Qin, C., You, H., Ran, H., and Fu, Y. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=3Pbra-_u76D.

Mao, J., Wang, X., and Li, H. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Maturana, D. and Scherer, S. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 922–928. IEEE, 2015.

Mialon, G., Chen, D., d'Aspremont, A., and Mairal, J. A trainable optimal transport embedding for feature aggregation and its relationship to attention, 2021.

Mohammadi, S. S., Wang, Y., and Bue, A. D. Pointviewgcn: 3d shape classification with multi-view point clouds. In *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 3103–3107, 2021. doi: 10.1109/ICIP42928.2021.9506426.

Moosmüller, C. and Cloninger, A. Linear optimal transport embedding: provable wasserstein classification for certain rigid transformations and perturbations. *Information and Inference: A Journal of the IMA*, 12(1):363–389, 2023.

Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv preprint arXiv:1811.01900*, 2018.

Naderializadeh, N., Comer, J. F., Andrews, R. W., Hoffmann, H., and Kolouri, S. Pooling by sliced-Wasserstein embedding. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=1z2T01DKEaE.

Prokudin, S., Lassner, C., and Romero, J. Efficient learning on point clouds with basis point sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017a.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017b. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

Qi, Z., Dong, R., Zhang, S., Geng, H., Han, C., Ge, Z., Wang, H., Yi, L., and Ma, K. Shapellm: Universal 3d object understanding for embodied interaction. In *European Conference on Computer Vision (ECCV)*, 2024.

Radenović, F., Tolias, G., and Chum, O. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7): 1655–1668, 2018.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Deep learning with sets and point clouds, 2017.

Ren, H., Dai, H., Dai, Z., Yang, M., Leskovec, J., Schuurmans, D., and Dai, B. Combiner: Full attention transformer with sparse computation cost. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 22470–22482. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/bd4a6d0563e0604510989eb8f9ff71f5-Paper.pdf.

Shen, Y., Feng, C., Yang, Y., and Tian, D. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4548–4557, 2018.

Skianis, K., Nikolentzos, G., Limnios, S., and Vazirgiannis, M. Rep the set: Neural networks for learning set representations. In *International conference on artificial intelligence and statistics*, pp. 1410–1420. PMLR, 2020.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, D. T., and Yeung, S.-K. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, Q., Xie, J., Zuo, W., Zhang, L., and Li, P. Deep cnns meet global covariance pooling: Better representation and generalization. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2582–2597, 2020a.

Wang, W., Slepčev, D., Basu, S., Ozolek, J. A., and Rohde, G. K. A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *International journal of computer vision*, 101:254–269, 2013.

Wang, W., Wang, R., Huang, Z., Shan, S., and Chen, X. Discriminant analysis on riemannian manifold of gaussian distributions for face recognition with image sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

Wang, Y., Hu, L., Wu, Y., and Gao, W. Graph multihead attention pooling with self-supervised learning. *Entropy*, 24(12), 2022. ISSN 1099-4300. doi: 10.3390/e24121745. URL https://www.mdpi.com/1099-4300/24/12/1745.

Wang, Z., Yao, K., Li, X., and Fang, S. Multi-resolution multi-head attention in deep speaker embedding. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6464–6468, 2020b. doi: 10.1109/ICASSP40776.2020.9053217.

Wang, Z., Yu, Y., and Li, X. Rethinking local-to-global representation learning for rotation-invariant point cloud analysis. *Pattern Recognition*, 154:110624, 2024. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2024.110624. URL https://www.sciencedirect.com/science/article/pii/S0031320324003753.

Wu, X., Lao, Y., Jiang, L., Liu, X., and Zhao, H. Point transformer v2: Grouped vector attention and partition-based pooling. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=I3mLa12s_H.

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

Xiang, T., Zhang, C., Song, Y., Yu, J., and Cai, W. Walk in the cloud: Learning curves for point clouds shape analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 915–924, October 2021.

Ye, H.-J., Hu, H., Zhan, D.-C., and Sha, F. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Yu, T., Meng, J., and Yuan, J. Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 186–194, 2018.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in Neural Information Processing Systems*, 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

Zhang, J., Chang, W.-C., Yu, H.-F., and Dhillon, I. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 7267–7280. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/3bbca1d243b01b47c2bf42b29a8b265c-Paper.pdf.

Zhang, Y., Hare, J., and Prügel-Bennett, A. Fspool: Learning set representations with featurewise sort pooling, 2020.

Zhang, Z., Hua, B.-S., and Yeung, S.-K. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

A. Point Cloud Classification Approaches

A.1. Blueprint of permutation invariant neural networks

In order to ensure that a function is $\mathcal{G}$-invariant, where $\mathcal{G}$ is a symmetry group, a consistent blueprint can be followed.

Symmetry group   A group G is defined with an operator $\circ : G \times G \to G$ if it satisfies associativity, closure under $\circ$, has an identity element e for $\circ$, and an inverse for each element $g \in G$. A symmetry group represents symmetries of a geometric object. In this work, we focus on symmetry groups of all permutations of a finite set, particularly in the context of point clouds. We denote by $\mathcal{G}$ the group of all permutations of the points in the point cloud.

$\mathcal{G}$-equivariant and $\mathcal{G}$-invariant   Given a signal $x(\omega)$ on a domain ω and a function f defined on x, f is $\mathcal{G}$-equivariant if it satisfies $f(\rho(g)x(\omega)) = \rho(g)f(x(\omega))$ for all $g \in \mathcal{G}$, where $\rho(g)$ is a geometric transformation belonging to $\mathcal{G}$. f is $\mathcal{G}$-invariant if it satisfies $f(\rho(g)x(\omega)) = f(x(\omega))$ for all $g \in \mathcal{G}$, meaning f is invariant to geometric transformations in $\mathcal{G}$.

General blueprint for permutation invariant neural networks   To create a permutation invariant network, we can construct a series of permutation equivariant layers followed by a global permutation invariant layer. A function f is considered permutation invariant if it satisfies $f(PX, PAP^T) = f(X, A)$, where $X \in \mathbb{R}^{N \times d}$ is a series of d-dimensional input vectors, $P \in \mathbb{R}^{N \times N}$ is a permutation matrix, and $A \in \mathbb{R}^{N \times N}$ is an adjacency matrix. We can express f as follows:

$$f = L_I \circ L_{e_m} \circ L_{e_{m-1}} \circ \cdots \circ L_{e_1} \tag{1}$$

where L_I is a global permutation invariant layer, which can be a pooling layer or any other type of permutation invariant layer. $\{L_{e_i}\}_{i=1}^m$ are permutation equivariant layers, and this blueprint is sufficient to make f invariant to permutation. Additional layers, such as local pooling layers, can be used to address the curse of dimensionality, as long as the entire composed function remains invariant with respect to input permutation.

A.2. Message passing

The message passing mechanism embeds node features in each layer by communicating with their neighbors. Let $G = (V, E, X)$ be a graph with nodes $V = \{v_1, v_2, \ldots, v_N\}$, edges $E \subseteq V \times V$, and feature matrix $X = \{x_1, x_2, \ldots, x_N\}$, where $x_i \in \mathbb{R}^d$ is the feature vector corresponding to node v_i. The adjacency matrix $A^{N \times N}$ denotes the presence of edges in the graph, where $[A]_{i,j} \in \{0, 1\}$.

Let $h_i^{(k)}$ be the hidden state of node i at hidden layer k, where $k = 0, \ldots, K$ and $h_i^{(0)} = x_i$. The message passed from node j to node i at iteration k is given by the learnable function $\text{Message}_\theta(\cdot, \cdot)$, which takes as input the hidden states of source node j and target node i:

$$m_{j \to i}^{(k)} = \text{Message}_\theta \left(h_j^{(k-1)}, h_i^{(k-1)} \right). \tag{2}$$

The aggregated message for node i is then computed by an Aggregation function from its neighboring nodes $\mathcal{N}_i$:

$$m_i^{(k)} = \text{Aggregation} \left(m_{j \to i}^{(k)} \right)_{j \in \mathcal{N}_i}. \tag{3}$$

Aggregation must be a locally permutation invariant operator, e.g., global average or summation. Finally, the updated hidden state of node i is obtained by applying the learnable function $\text{Update}_\theta(\cdot, \cdot)$ to the previous hidden state and the aggregated message:

$$h_i^{(k)} = \text{Update}_\theta \left(h_i^{(k-1)}, m_i^{(k)} \right). \tag{4}$$

This mechanism learns helpful representations at the last layer, K, and uses them for any downstream tasks.

A.3. Learning from sets

Sets are unordered collections of feature vectors, e.g, point clouds and LiDAR scans. Learning from sets is akin to learning from graph-structured data, as sets can be considered as edge-less graphs or fully connected graphs. In the former, each node performs the message passing mechanism on itself, while in the latter, all nodes communicate with each other. This core

principle also underlies transformer networks, where similarity between queries and keys determines the attention weights that control information flow.

Considering sets as a fully-connected graph, the message passing is the same as in equations 2, 3, and 4. However, considering them as edge-less nodes of a graph results in no communication between nodes. As a result, equation 2 will become $m_i^{(k)} = \text{Message}_\theta \left(h_i^{(k-1)} \right)$. Consequently, the aggregated message will be calculated as $m_i^{(k)} = \text{Aggregation} \left(m_i^{(k)} \right)$, and the hidden state will be updated only by the node itself.

B. Backbone and Pooling Hyperparameters

B.1. Backbones

For the MLP backbone, representing the DeepSets architecture, we use 2 hidden layers of 512 neurons and an output layer of dimension 3. We use Leaky ReLU activation on the hidden layers.

For both the SAB and ISAB backbones in Backbone vs Pooling experiment, we use 1 hidden layer of dimension 256 and 4 heads. Furthermore, for the ISAB backbone, we use 16 inducing points in the architecture.

We use the original backbone implementation without modification of DGCNN, PointNet, and CurveNet, as published by the respective authors of each backbone.

B.2. Poolings

For the Generalized Mean pooling, we use a moment of 2. For the k-Max pooling, we use $k = 2$. For PMA we use 1024 seed vectors with just 1 head. For both L-PSWE and F-PSWE we use 1024 reference points and 1024 projections. For FSPool, we use 1024 pieces in the piecewise linear function and relaxed sorting.

For both GMHA and MMHA, we use 4 heads, while we use 2 affine layers for GMHA and 1 affine layer for MMHA. We do not use the temperature hyperparameter.

Finally, for ApprRepSet, we use 3 hidden sets with 5 elements.

C. Depth vs. Width Results for SAB Backbone

We observed that the SAB backbone, which we consider to be a more "complex" backbone, performs well in general, regardless of the pooling method in use. This is consistent with previously observed results, which showed that the benefit of particular pooling layers, such as the OT-based and attention-based poolings, diminished with increasing backbone complexity. Furthermore, as expected, increasing the depth and width beyond certain limits leads to a decrease in the test accuracy.
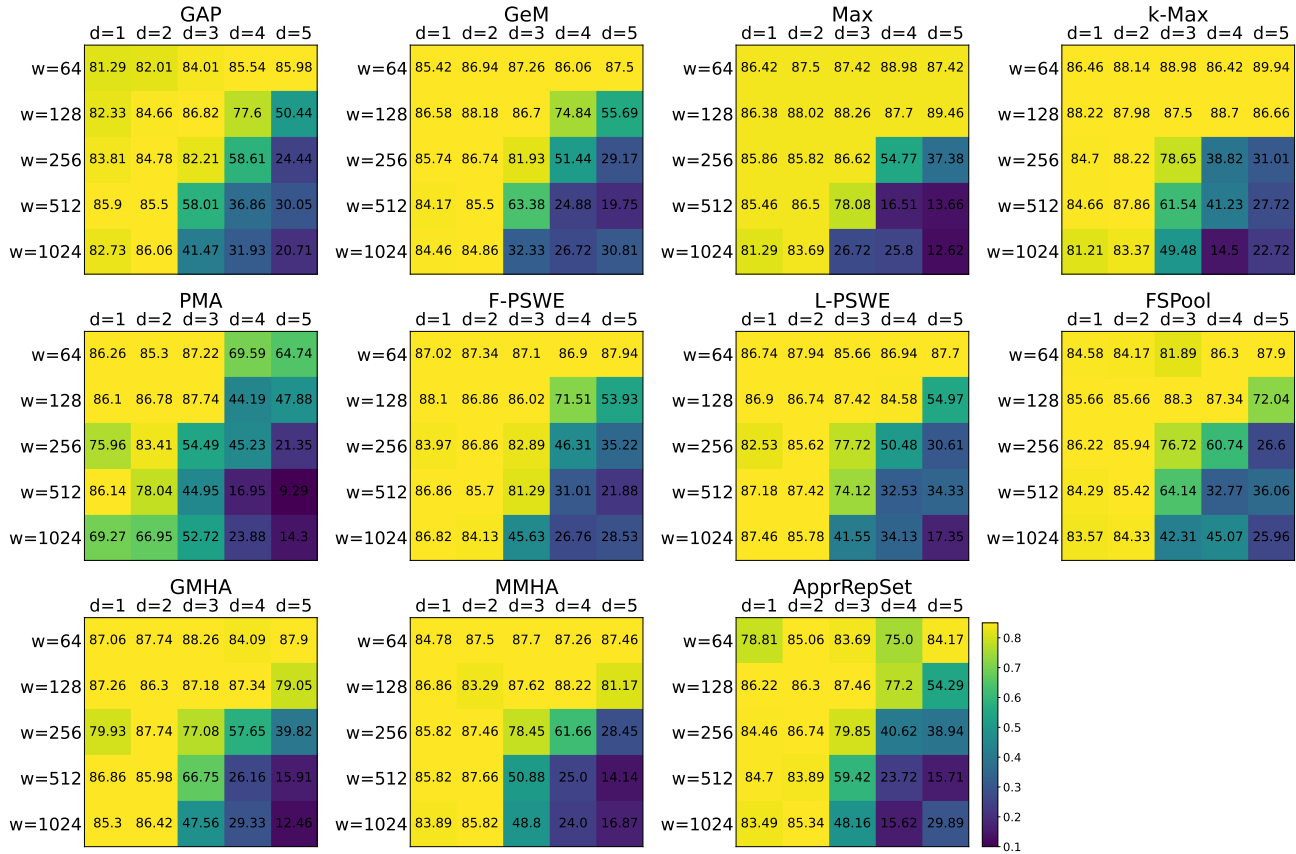
Figure 6: **Depth vs. Width.** The performance of each pooling method on SAB backbones with varied widths and depths is illustrated in the figure.

D. Paired Pooling Results on ScanObjectNN and ShapeNetPart

D.1. ScanObjectNN

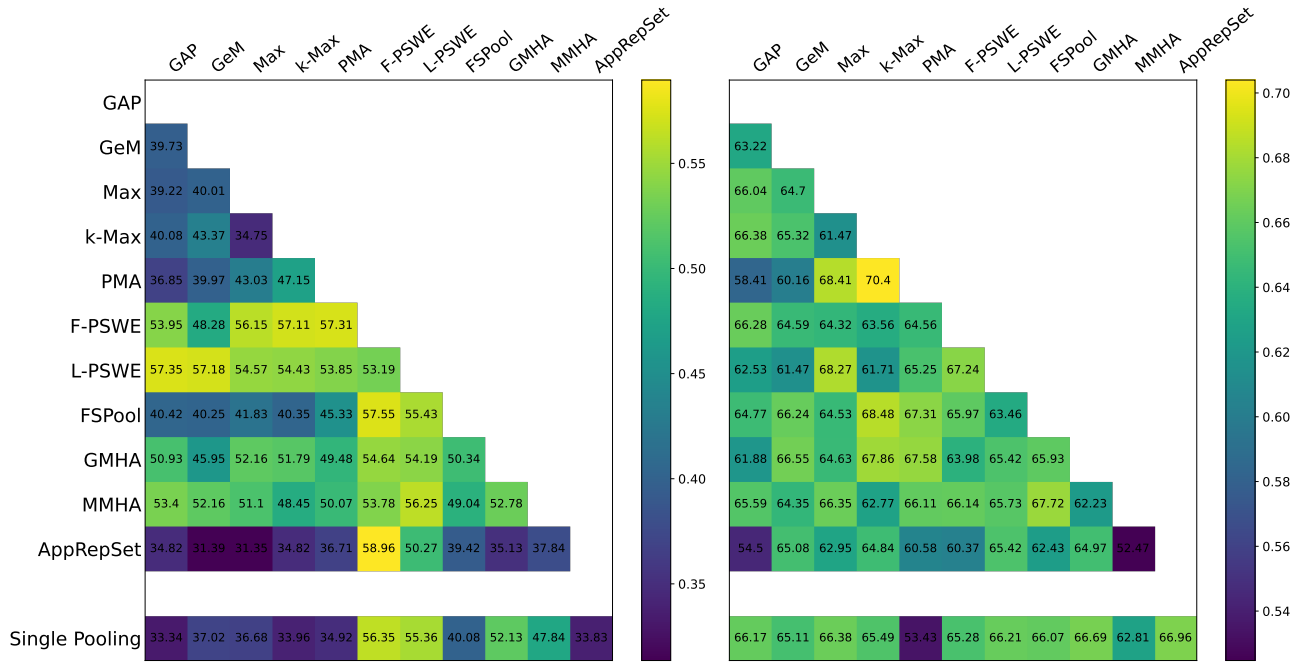Figure 7: **Effect of Paired Pooling Methods on ScanObjectNN.** Test set classification accuracy for models trained with the same backbone and different pairs of pooling layers. Results are shown for MLP (left) and SAB (right) backbones. The bottom rows in both plots represent the performance of the same model with a single pooling layer.

D.2. ShapeNetPart

MLP backbone (left)

	GAP	GeM	Max	k-Max	PMA	F-PSWE	L-PSWE	FSPool	GMHA	MMHA	AppRepSet
GAP											
GeM	92.81										
Max	92.6	92.74									
k-Max	91.94	92.56	86.7								
PMA	94.44	93.05	91.76	93.88							
F-PSWE	96.75	96.38	96.93	96.28	96.2						
L-PSWE	96.28	96.62	96.9	96.69	96.17	97.01					
FSPool	92.55	93.14	92.97	92.34	94.81	96.21	96.45				
GMHA	95.93	95.83	95.34	95.17	95.65	96.94	97.31	96.66			
MMHA	94.02	95.96	94.53	96.35	95.37	96.03	97.21	95.54	96.31		
AppRepSet	86.07	87.46	82.24	84.47	91.67	95.72	96.27	93.6	86.82	89.33	
Single Pooling	88.33	87.59	79.59	80.96	93.52	97.25	96.13	92.58	94.95	95.79	85.37

SAB backbone (right)

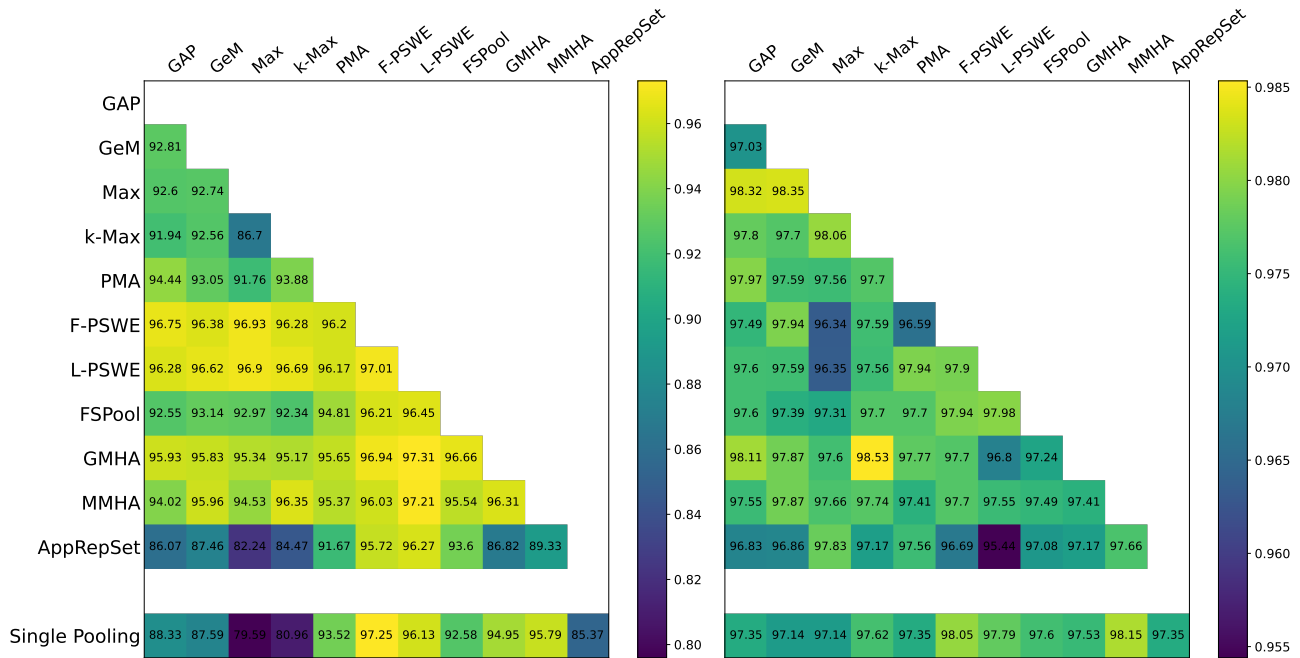|	GAP	GeM	Max	k-Max	PMA	F-PSWE	L-PSWE	FSPool	GMHA	MMHA	AppRepSet
GAP											
GeM	97.03										
Max	98.32	98.35									
k-Max	97.8	97.7	98.06								
PMA	97.97	97.59	97.56	97.7							
F-PSWE	97.49	97.94	96.34	97.59	96.59						
L-PSWE	97.6	97.59	96.35	97.56	97.94	97.9					
FSPool	97.6	97.39	97.31	97.7	97.7	97.94	97.98				
GMHA	98.11	97.87	97.6	98.53	97.77	97.7	96.8	97.24			
MMHA	97.55	97.87	97.66	97.74	97.41	97.7	97.55	97.49	97.41		
AppRepSet	96.83	96.86	97.83	97.17	97.56	96.69	95.44	97.08	97.17	97.66	
Single Pooling	97.35	97.14	97.14	97.62	97.35	98.05	97.79	97.6	97.53	98.15	97.35

Figure 8: **Effect of Paired Pooling Methods on ShapeNetPart.** Test set classification accuracy for models trained with the same backbone and different pairs of pooling layers. Results are shown for MLP (left) and SAB (right) backbones. The bottom rows in both plots represent the performance of the same model with a single pooling layer.

As demonstrated in Figures 7 and 8, combining specific pooling methods can enhance performance, though we once again observe that not all poolings are compatible. In the case of ScanObjectNN, we find that, while the OT-based and attention-based poolings generally perform the best in the single pooling case, pairing k-Max pooling with PMA pooling boosts test accuracy to 70.4% with the SAB backbone. When considering ShapeNetPart, we observe that pairing L-PSWE with the attention-based GMHA and MMHA poolings can result in a performance boost with the MLP backbone. These experiments highlight that pairing certain complementary poolings at the final permutation invariant layer can lead to notable performance improvements.