Few-Shot Learning for 3D Shape Classification

Rishabh Gupta
UMass Amherst
rishabhgupta@umass.edu

Abstract

The ability of deep neural networks to learn high level representations from large labelled datasets is proven. Yet current deep learning approaches suffer from poor sample efficiency in stark contrast to human perception. Few-shot learning algorithms mitigate this problem, by learning a model to recognize previously unseen classes with limited labeled examples. Recent deep learning methods for classifying 3D shapes have achieved over 90% accuracy on several standard datasets, including ModelNet40 and ModelNet10. Moreover, due to significant progress in depth camera technology, it is now possible to capture 3D data at scale much more easily. It is therefore likely that many classes of 3D objects will not be present in the labeled training set. As a result, few-shot classification systems will be needed to leverage other more easily-obtainable sources of information in order to classify unseen objects. Even though there is plenty of recent work which explore few-shot learning for 2D image data, the corresponding 3D shape classification problem has not been meaningfully explored in few-shot regime. In this project, we present a consistent comparative analysis of different 3D representations using representative few-shot learning algorithms. We construct K-shot, N-way classification tasks from ModelNet40 dataset and train meta-learners for 3D shape classification task. We also analyze cross-domain generalization ability of our meta-trained models on support sets from McGill and SHREC2015 datasets. Our results reveal that multi-view representations perform best for few-shot classification but the performance gap with point clouds is smaller compared to standard supervised learning setting. In the cross-domain evaluation setting, we show that a transfer learning based baseline method outperforms the state-of-the-art few-shot learning algorithms.

1. Introduction

Deep learning has shown tremendous success in the past few years on various visual recognition tasks. But this success can be majorly attributed to the availability of large number of labeled examples. Meanwhile, the recent introduction of depth sensors has made 3D data collection easier. With the proliferation of 3D data, it has become difficult and expensive to annotate data. The annotation cost as well as the scarcity of data in some classes significantly limit the applicability of deep learning techniques to 3D vision tasks. In contrast, the human visual systems can recognize new classes with extremely few labeled examples. Motivated by the sample efficiency of humans, few-shot learning techniques [4, 15] have recently become popular to learn models that could generalize well to previously unseen classes with very few labeled examples.

In few-shot classification [3], a classifier must be adapted to accommodate new samples which are previously unseen, given only a few examples of their classes at training time. While the problem is quite difficult, it has been demonstrated that humans have the ability to perform even one-shot classification, where only a single example of each new class is given, with a high degree of accuracy. A naive approach, such as pretraining and fine-tuning the model on the new data, would severely overfit in this case. One promising direction to few-shot classification is the meta-learning paradigm where transferable knowledge is extracted and propagated from a collection of tasks to prevent overfitting and improve generalization. In meta-learning, the goal of the trained model is to quickly learn a new task from a small amount of new data and the model is trained by the meta-learner to be able to learn on a large number of different tasks. While such methods are typically trained on a set of so-called "seen" classes, they are capable of classifying "unseen" classes with very few labeled examples. The defacto standard for few shot learning is K-shot,N-way classification tasks which can be briefly described as follows: a model is given a query sample belonging to a new, previously unseen class. It is also given a support set S consisting of K examples each from N different unseen classes. The algorithm then has to determine which of the support set classes the query sample belongs to.

In this work, we focus on the problem of few shot learning for 3D shape classification and present a detailed empirical study to shed new light on the few-shot classifica-

tion problem from a 3D perspective. Specifically, we conduct consistent comparative experiments to compare different 3D representations [17] in few-shot regime using representative meta-learning algorithms like Prototypical Networks (ProtoNets) [15] and Model-Agnostic Meta Learning (MAML) [4]. The performance of few-shot approaches is also compared with a simple transfer learning baseline. Also, our evaluation not only focuses on recognizing novel classes sampled from the base dataset ModelNet40 [23] but also to unseen shapes drawn from an entirely different datasets like SHREC2015 [8] and McGill [14]. Third, we introduce a practical evaluation setting where there exists domain shift between base and novel classes (e.g., sampling base classes from everyday rigid object categories and novel classes from non-rigid objects). Our results show that sophisticated few-shot learning algorithms do not provide performance improvement over the baseline under this setting.

Our contributions:

1. We provide a comparative analysis of different 3d representations in few-shot regime. We find that MVCNN [16] performs the best for few-shot classification but the performance gap with PointNet [10] is smaller as compared to standard supervised classification setting.
2. Our empirical evaluation results on MVCNN reveal that in few-shot setting, performance improvements obtained from using multiple shots with single views are smaller as compared to single instance with multiple views.
3. We investigate a practical evaluation setting where base and novel classes are sampled from different domains (rigid vs non-rigid shapes). We show that current few-shot classification algorithms fail to address such domain shifts and are inferior even to the transfer learning baseline, highlighting the importance of learning to adapt to domain differences in few-shot learning.

The rest of the report is organized as follows. In Section 2 , we introduce the related work; the proposed approach will be introduced in Section 3 ; in Section 4, the datasets and the related experiments will be discussed. We also will discuss the related experimental results in this section; finally, the conclusion will be provided in Section 5.

2. Background/Related Work

2.1. 3D Shape Classification

A large number of shape descriptors have been developed for drawing inferences about 3D objects in computer vision and graphics literature. Researchers usually classify shape descriptors into two broad categories: native 3D shape descriptors and view-based descriptors. The first one directly works on the native 3D representations of objects, for example, polygon meshes, voxel-based discretizations, point clouds, or implicit surfaces; the second one describes the shape of an 3D object by how it looks in a collection of 2D projections. Previous 3D shape descriptors were largely hand-designed according to a particular geometric property of the shape surface or volume. The early methods utilizing deep learning for operating on 3D data used volumetric or multi-view representations in order to work with 3D data. Recently, the trend in this area has shifted to instead using raw point clouds directly, without any preprocessing step. However, applying deep learning on 3D datasets is typically challenging because the size of organized databases with annotated 3D models is rather limited compared to 2D image datasets.

Volumetric methods: These methods [9] learn shape descriptors from the voxel-based representation of an object through 3D convolutional networks.

Multi-View based methods: MVCNN [16] projects a 3D object into multiple views, extracts view-wise CNN features and max-pools them into a global representation of the 3D object. GIFT [1] also extracts view-wise features but do not pool them. Instead, it obtains the similarity between two 3D objects by view-wise matching. More recently, [20] recurrently cluster the views into multiple sets, pool the features in each set and achieve better performance than the original MVCNN.

Point Cloud based methods These methods directly take unordered point sets as inputs addressing the sparsity problem encountered in volumetric methods, and they do not make any a priori assumptions onto which 2D planes, and how many, that the point cloud should be projected on, like the view-based methods do. PointNet [10] was the first work that operated on raw point clouds directly at the input of the network. PointNet used a multi-layer perceptron (mlp) to extract features from point sets,and max-pooling layers to incorporate permutation invariance. Recently, [11] improve PointNet by exploiting local structures induced by the metric space.

2.2. Few-shot Learning

Given abundant training examples for the base classes, few-shot learning algorithms aim to learn to recognizing novel classes with a limited amount of labeled examples. In the following, we discuss representative few-shot learning algorithms organized into three main categories: initialization based, metric learning based, and hallucination based methods.

Initialization based methods tackle the few-shot learning problem by "learning to fine-tune". Model-Agnostic Meta-Learning (MAML) [4] and related approaches aim to learn good model initialization so that the classifiers for novel classes can be learned with a limited number of labeled examples and a small number of gradient update steps. Another line of work focuses on learning an optimizer. Examples include the LSTM-based meta-learner

for replacing the stochastic gradient decent optimizer [12] and the weight-update mechanism with an external memory. While these initialization based methods are capable of achieving rapid adaption with a limited number of training examples for novel classes, previous experiments show that these methods have difficulty in handling domain shifts between base and novel classes. In this work, we primarily focus on MAML.

Distance metric learning based methods address the few-shot classification problem by "learning to compare". The intuition is that if a model can determine the similarity of two images, it can classify an unseen input image by comparing it with labeled instances [7]. To learn a sophisticated comparison models, meta-learning based methods make their prediction conditioned on a distance metric to few labeled instances during the training process. Examples of distance metrics include cosine similarity [19], Euclidean distance to class-mean representation [15], CNN-based relation module [18], ridge regression [2], and graph neural network [5]. In this work, we primarily focus on prototypical networks (ProtoNets).

Hallucination based methods directly deal with data deficiency by "learning to augment". This class of methods learns a generator from data in the base classes and use the learned generator to hallucinate new novel class data for data augmentation. One type of generator aims at transferring appearance variations exhibited in the base classes. These generators either transfer variance in base class data to novel classes [6], or use GAN models to transfer the style. Another type of generators [21] does not explicitly specify what to transfer, but directly integrate the generator into a meta-learning algorithm for improving the classification accuracy. Since hallucination based methods often work with other few-shot methods together (e.g. use hallucination based and metric learning based methods together) and lead to complicated comparison, we do not include these methods in our comparative study and leave it for future work.

2.3. Few-shot Learning on 3D data

Since it is easier to do end-to-end learning with raw point clouds and data annotation problem is most encountered in more challenging tasks like 3D object detection or segmentation, most of the recent work in this space [13, 22] focuses on few-shot learning with point clouds on such tasks. In this work, our objective is to have a deeper look at how different 3D representations perform in few-shot regime and how is it different from standard supervised regime. For simplicity, we pick the standard task of 3D shape classification. But it is an interesting direction to explore how the results change for more challenging 3D tasks.

3. Approach

3.1. Problem Setup

In the standard supervised learning setting, we split D to optimize parameters on a training set D_{train} and evaluate its generalization on the test set D_{test}. However, in the few-shot setting, we split D into multiple datasets called meta-sets for meta-training ($D_{meta-train}$), meta-validation ($D_{meta-valid}$), and meta-testing ($D_{meta-test}$) such that there is no overlap of classes between different meta-sets.

We divide each meta-set into episodes and each episode into a training set D_{train} and a test set D_{test}. The k-shot and the N-way classification task is considered. In each episode, D_{train} consists of k labelled examples for each of N classes, and D_{test} has a fixed number of examples for evaluation. According to the design, we can perform hyper-parameter selection of the meta-learner using $D_{meta-valid}$ and evaluate its generalization performance on $D_{meta-test}$. Figure 1 visualizes an example split for ModelNet40 dataset.

3.2. Model Architectures

Here, we describe the representative model architectures of different 3D representations that we compare in our study on 3D shape classification.

- **Multi-View CNN (MVCNN)**: The MVCNN architecture [16] uses rendered images of the model from different views as input. Each image is fed into a CNN with shared weights. A max-pooling layer across different views is used to perform an orderless aggregation of the individual representations followed by several non-linear layers for classification. For reporting results, we use the VGG-11 network as the feature backbone.

- **VoxNet**: The VoxNet [9] was first proposed in several early works that uses convolution and pooling layers defined on 3D voxel grids. The early VoxNet models used two 3D convolutional layers and 2 fully-connected layers. We use a deeper VoxNet architecture which has five blocks of (conv3d-batchnorm-LeakyReLU) and includes batch normalization. All conv3d layers have kernel size 5, stride 1 and channel size 32. The Leaky ReLU has slope 0.1. Two fully-connected layers(fc-batchnorm-ReLU-fc) are added on top to obtain class predictions.

- **PointNet**: PointNet [10] is a novel system for extracting both local and global features from point cloud data. The architecture applies a series of non-linear mappings individually to each input point and performs orderless aggregations using max-pooling operations. Thus the model is invariant to the order in

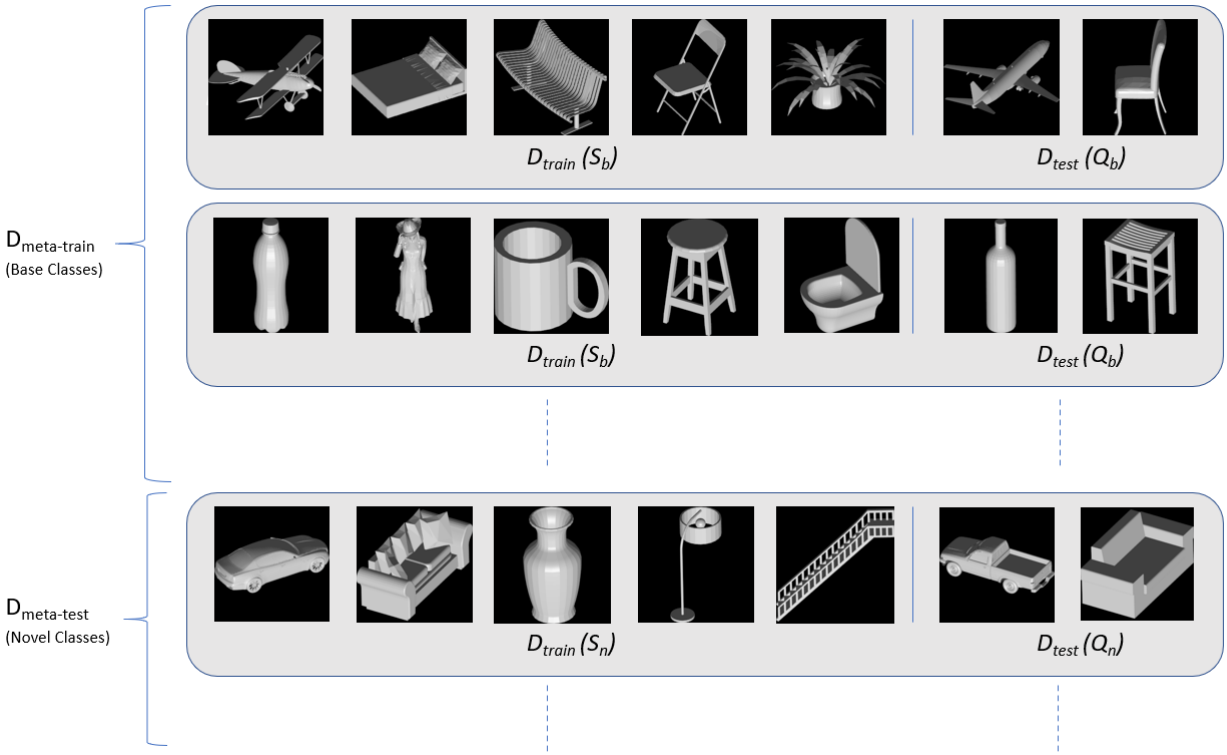

Figure 1. Meta-learning setup for 3D shape classification with an example 1-shot, 5-way task split for ModelNet40

which the points are presented and can directly operate on point clouds without additional preprocessing such as spatial partitioning, or graph construction. The input and feature transformations utilize a T-net which learns an affine transformation matrix. The output of the classification network is an N-way classification score.

3.3. Transfer Learning Baseline

This baseline model follows the standard transfer learning procedure of network pre-training and fine-tuning. During training stage, we train the above mentioned feature extractors and a classifier from scratch by minimizing a standard cross-entropy classification loss using the training examples in the base classes. During fine-tuning stage, we fix the pre-trained network parameter in our feature extractor and train a new classifier by minimizing the loss using the few labeled examples (i.e.,the support set) in the novel classes.

3.4. Meta-Learning Algorithms

Here, we describe in detail the formulations of representative meta-learning algorithms used in our work. Meta-learning algorithms consist of a meta-training and a meta-testing stage. In the meta-training stage, for each episode, the objective is to train a classification model M that minimizes $N-$way prediction loss of the samples in the query set Q_b. Here, the classifier is conditioned on provided support set S_b. By making prediction conditioned on the given support set, a meta-learning method can learn how to learn from limited labeled data through training from a collection of tasks (episodes). In the meta-testing stage, all novel class labeled data S_n are considered as the support set for novel classes, and the classification model can be adapted to predict novel classes from the new query set Q_n. This new notations are visualized in Figure 1. Different meta-learning methods differ in their strategies to make prediction conditioned on support set. Figure 2 explains the following meta-learning algorithms used in our work.

- **Prototypical Networks (ProtoNets) [15]**: This is a metric-learning based meta-learning algorithm. For ProtoNet, the prediction of the examples in a query set is based on comparing the distance between the query feature and the support feature from each class. Specifically, ProtoNet compares the Euclidean distance between query features and the class mean of support features.

- **Model-Agnostic Meta-Learning (MAML) [4]**: This is an initialization based meta-learning algorithm, where each support set is used to adapt the initial model parameters using few gradient updates. As dif-

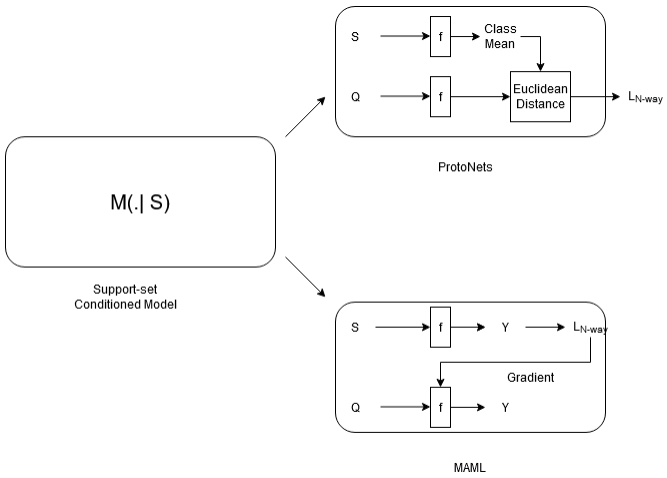

Figure 2. Meta-learning algorithms used in our work

ferent support sets have different gradient updates, the adapted model is conditioned on the support set. Note that when the query set instances are predicted by the adapted model in the meta-training stage, the loss of the query set is used to update the initial model, not the adapted model.

4. Experiments

In this section we describe the datasets we used, implementation and training details, our experimental results and finally the analysis of results.

4.1. Datasets

We evaluate our approach on four well-known 3D datasets, ModelNet10 [23], ModelNet40 [23], McGill [14], and SHREC2015 [8]. The ModelNet40 contains 12,311 CAD models from 40 different classes, and ModelNet10, which is a subset of ModelNet40, consists of 4899 CAD models from 10 different classes. The ModelNet40 contains 9,843 training samples and 2,468 testing samples, and ModelNet10 includes 3991 training samples and 908 testing samples. We use McGill and SHREC2015 datasets for testing cross-domain generalization, since they contain non-rigid shapes which are different from rigid shapes seen in ModelNet40. Moreover, the McGill dataset consists of 456 CAD models of 19 different objects. The SHREC2015 dataset consists of 1200 3D watertight triangle meshes from 50 different classes, where each class contains 24 objects with distinct postures.

Seen/Unseen Split: We use a suitable and standard split of seen and unseen classes similar to the 2D version of the corresponding few-shot problem [3]. The idea of using a single split is better than using several random splits as it establishes a common testbed for evaluating other methods. In a real life setting, unseen 3D point cloud objects are likely obtained from an advanced 3D-depth camera or laser scanner. Here, however, we attempt to simulate that scenario by using 3D point cloud objects from a different dataset not included during training. Therefore, we propose to use 30 classes from ModelNet40 as seen and other, disjoint classes, from ModelNet10, McGill and SHREC2015 as unseen. In ModelNet10, which has 10 classes, only testing samples based on the splitting protocol mentioned above, are considered as the unseen set. In McGill, 5 classes, which also appear in the seen set, are removed, and the remaining 14 categories are selected. Since we follow the protocol introduced by, we only consider their testset, the last third of the instances, as the unseen in-stances to enable a fair comparison. Finally, in the SHREC2015 dataset, those classes that were similar to the seen classes were removed. The remaining 30 of the original 50 classes were chosen as the unseen instances. Moreover, in line with the protocol used in, 25% of instances were chosen randomly as unseen samples. To summarize, we follow the seen/unseen splits where the seen classes are those 30 in ModelNet40 that do not occur in ModelNet10, and the unseen classes are those from the test sets of ModelNet10, McGill and SHREC2015 that are not in the set of seen classes. These splits allow us to test unseen classes from different distributions than that of the seen classes. The dataset statistics as used in this work are summarized in Table 1.

Episode Creation: Under k-shot, N-way classification setting, many related but small training sets of k examples for each of N classes are treated as the input of our method. In order to generate each instance of a k-shot, N-way task dataset $D = (D_{train}, D_{test}) \in D$, the following steps is performed: first, N classes from the list of classes is sampled corresponding to relative the meta-set; then k examples is sampled from each of those classes. The training set D_{train} is consisted of these k examples together. Besides, we sample an additional fixed amount of the rest of the examples to yield a test set D_{test}. Generally there are 15 examples per class in the test sets. When we train the meta-learner, these datasets (episodes) are iterated repeatedly by sampling. As for meta-validation and meta- testing, however, a fixed number of these datasets is produced to evaluate each method. Enough datasets are produced to ensure that we shall maintain the confidence interval of the mean accuracy to be small.

4.2. Implementation Details

In this subsection, we specify the implementation details of our experiments.

In the training stage for the baseline method, we train 400 epochs with a batch size of 16. In the meta-training stage for meta-learning methods, we train $60,000$ episodes for $1-$shot and $40,000$ episodes for $5-$shot tasks. We use

Dataset	Total Classes	Seen/Unseen	Train/Val/Test
ModelNet40	40	$30/-$	$5852/1560/-$
ModelNet10	10	$-/10$	$-/-/908$
McGill	19	$-/14$	$-/-/115$
SHREC2015	50	$-/30$	$-/-/192$

Table 1. Statistics of 3D datasets used

the validation set to select the training episodes with the best accuracy. In each episode, we sample N classes to form N-way classification (N is 5 in both meta-training and meta-testing stages unless otherwise mentioned). For each class, we pick k labeled instances as our support set and 16 instances for the query set for a $k-$shot task.

In the fine-tuning or meta-testing stage for all methods, we report the mean of 600 randomly generated test episodes. In each experiment, we randomly sample 5 classes from novel classes, and in each class, we also pick k instances for the support set and 16 for the query set. For transfer learning baseline, we use the entire support set to train a new classifier for 100 iterations with a batch size of 4. For meta-learning methods, we obtain the classification model conditioned on the support set.

All methods are trained from scratch and use the Adam optimizer with initial learning rate 10^{-3}. For MAML, we use a first-order approximation in the gradient for memory efficiency. We implemented our models in PyTorch and used a NVIDIA GTX1080TI GPU to train the models.

We now conduct experiments on the most common setting in few-shot classification, 1-shot and 5-shot classification, i.e., 1 or 5 labeled instances are available from each novel class. We use VoxNet, MVCNN and PointNet described in Section 3 as backbones depending on the input 3D representations and perform 5-way classification for only novel classes during the fine-tuning or meta-testing stage.

We used Monte Carlo cross-validation over the meta-validation set to find the best hyper-parameters, averaging over 10 repetitions. For ModelNet40, 20% (6) of the 30 seen classes were randomly selected as an unseen validation set.

4.3. Results and Discussion

In this section, the results of experiments are described, the performance of different approaches including VoxNet, PointNet and MVCNN is compared against each other under few-shot setting using two meta-learning algorithms and a transfer learning baseline. The experimental results of generalization to ModelNet10, McGill, SHREC2015 are summarized in Table 2.

Fewshot Adaptation to Unseen Classes: We observe that the difference in accuracy between few-shot learning and supervised learning is still very large for 3D shape classification, e.g. 53% as compared to 95.7% for ModelNet10 reported earlier [17]. As such, there is significant potential for improvement for few-shot 3D point cloud classification. We also observe that the performance gap between 1-shot and 5-shot results is smaller in 3D setting as compared to 2D setting as reported previously because a single shot of a 3D shape provides enough information about the geometry of the given object which is not the case in 2D image classification.

Comparison of 3D classification architectures: In the few-shot classification setup, we observe that overall MVCNN perform the best. However, the performance gap between MVCNN and PointNet is smaller as compared to the standard supervised setting. We think the reason behind this might be that in the current few-shot setup, we train all the backbones from scratch for fair comparison and MVCNN can no longer benefit from pre-training from large amounts of 2D image data. Both methods, however, outperform VoxNet architecture.

Comparison of Training Methods: We observe that transfer learning is inferior to meta-learning algorithms which are more effective in generalizing to unseen classes. Between Protonets and MAML, we observed that MAML works better for 1-shot tasks but ProtoNet works better for 5-shot tasks.

Performance of MVCNN with different number of views and shots: For the case of MVCNN, we also analyze the effect of different number of views on the performance on ModelNet10. From Table 3, we observe that we can tradeoff 5 shots of a given shape with single view with a single-shot of the same shape with 5 views. This means 3D data is inherently more suited for few-shot classification since using multiple views from a single shot is provides more information about the object geometry than using multiple shots with single views.

Per-class results: In Figure 3, we report individual class performance of unseen ModelNet10 classes for MVCNN. Our architecture performs better on ModelNet10 classes (e.g., sofa, table, dresser, and bathtub) for which there are sufficiently similar classes in the seen set. However, one can find that the architecture cannot classify some unseen classes at all (for example toilet, crab, dinosaur etc). We also observe that while comparing 1-shot and 5-shot results of different unseen classes, the more diverse shapes (like desk, chair, sofa, table) have more gap in the 5-shot and 1-shot results whereas other shapes which look very similar across instances have little benefit from using multiple shots.

Cross-domain Generalization: Below we discuss how domain differences between base and novel classes impact few-shot classification results. As shown in Table 2, the transfer learning baseline outperforms both meta-learning

Model	ModelNet10		McGill		SHREC2015	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Pre-trained VoxNet + Fine-tuning	37.56	49.43	25.6	31.1	15.7	23.9
Pre-trained MVCNN (x12) + Fine-tuning	39.86	52.15	27.4	32.7	16.4	24.5
Pre-trained PointNet + Fine-tuning	39.12	51.87	27.4	32.6	17.5	25.2
VoxNet + ProtoNets	42.37	52.73	20.9	28.6	13.0	21.3
MVCNN (x12) + ProtoNets	43.32	54.08	23.7	30.7	14.2	22.0
PointNet + ProtoNets	43.21	53.87	23.3	30.4	15.0	23.1
VoxNet + MAML	43.40	51.54	20.1	27.8	11.7	20.6
MVCNN (x12) + MAML	44.90	52.99	22.4	29.5	12.9	21.5
PointNet + MAML	44.12	52.94	21.7	29.2	13.0	21.9

Table 2. Few-shot Classification results of models trained on ModelNet40 on unseen classes from ModelNet10, McGill and SHREC2015

Model	ModelNet10		
	Num Views	1-shot	5-shot
MVCNN + ProtoNets	1	35.45	39.87
MVCNN + ProtoNets	5	41.12	52.11
MVCNN + MAML	1	36.28	41.11
MVCNN + MAML	5	42.32	52.01

Table 3. Few-shot Classification results for MVCNN on ModelNet10 using different number of views

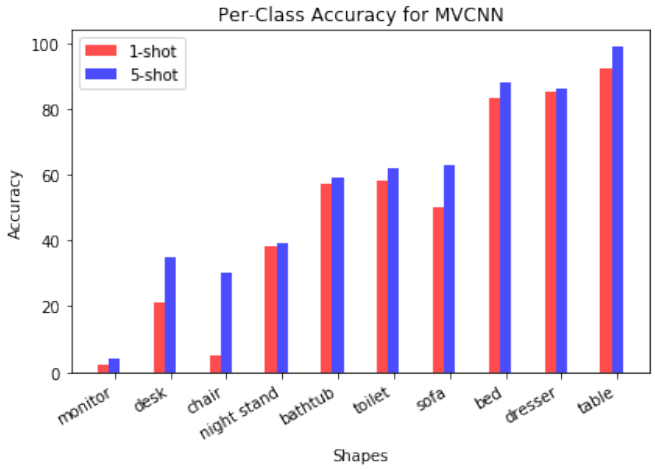

Figure 3. Per-class Accuracy for MVCNN with ProtoNets on ModelNet10

methods under this scenario. While meta-learning methods learn to learn from the support set during the meta-training stage, they are not able to adapt to novel classes that are too different since all of the base support sets are within the same dataset. In contrast, the baseline simply replaces and trains a new classifier based on the few given novel class data, which allows it to quickly adapt to a novel class and is less affected by domain shift between the source and target domains.

Our meta-learning methods do not perform as well on the McGill and SHREC2015 datasets when compared to the ModelNet10 results, because the shapes of unseen McGill and SHREC2015 datasets are significantly different from the distribution in the seen ModelNet40 dataset (non-rigid vs rigid), much more so than that of ModelNet10. To further dig into the issue of domain difference, we design scenarios that provide such domain shifts. We believe that this is practical scenario since collecting shapes from a everyday rigid objects may be relatively easy (e.g.due to increased availability) but collecting images from non-rigid classes might be more difficult. Moreover, such non-rigid classes may have large diversity across instances as compared to rigid shapes in ModelNet10.

To further adapt meta-learning methods as in the baseline method, an intuitive way is to fix the features and train a new softmax classifier. However, changing the setting in the meta-testing stage can lead to inconsistency with the meta-training stage. Thus, we believe that learning how to adapt in the meta-training stage is important future direction.

5. Conclusion

Traditional recognition systems have achieved superior performance on 3D objects. However, due to the advancement of 3D depth camera technology, obtaining 3D datasets has become much more accessible than before. Hence, we will more than likely encounter many unseen objects not encountered during training. Therefore, it is time for 3D recognition systems to adapt few-shot setting, aiming to recognize those unseen objects. To this end, this work described a comparative study for pushing forward with this line of investigation by comparing different 3D representations in few-shot regime. We modified some established 3D shape classification architectures to work in the few-shot setting using recent state-of-the-art meta-learning algorithms like MAML and ProtoNets. Through comparing methods on a common ground, our results also show that a transfer learning baseline achieves competitive perfor-

mance with meta-learning algorithms under a realistic scenario where there exists domain shift between the base and novel classes. It still needs to be seen how to improve meta-learning algorithms to adapt to such out-of-domain classes. Also, a similar study is needed to evaluate what 3D representations work best for more challenging tasks like 3D object detection and segmentation in a few-shot setting.

References

[1] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Latecki. Gift: A real-time and scalable 3d shape search engine. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5023–5032, 2016.

[2] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[3] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[4] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1126–1135. JMLR.org, 2017.

[5] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[6] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[7] G. R. Koch. Siamese neural networks for one-shot image recognition. 2015.

[8] Z. Lian, J. Zhang, S. Choi, H. ElNaghy, J. El-Sana, T. Furuya, A. Giachetti, R. A. Guler, L. Lai, C. Li, H. Li, F. A. Limberger, R. Martin, R. U. Nakanishi, A. P. Neto, L. G. Nonato, R. Ohbuchi, K. Pevzner, D. Pickup, P. Rosin, A. Sharf, L. Sun, X. Sun, S. Tari, G. Unal, and R. C. Wilson. Non-rigid 3D Shape Retrieval. In I. Pratikakis, M. Spagnuolo, T. Theoharis, L. V. Gool, and R. Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2015.

[9] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[11] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[12] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[13] G. Sharma, E. Kalogerakis, and S. Maji. Learning point embeddings from shape repositories for few-shot segmentation. In *2019 International Conference on 3D Vision (3DV)*, pages 67–75. IEEE, 2019.

[14] K. Siddiqi, J. Zhang, D. Macrini, A. Shokoufandeh, S. Bouix, and S. Dickinson. Retrieving articulated 3-d models using medial surfaces. *Machine Vision and Applications*, 19:261–275, 2007.

[15] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4080–4090, Red Hook, NY, USA, 2017. Curran Associates Inc.

[16] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[17] J.-C. Su, M. Gadelha, R. Wang, and S. Maji. A deeper look at 3d shape classifiers. *ArXiv*, abs/1809.02560, 2018.

[18] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[19] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[20] C. Wang, M. Pelillo, and K. Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. *arXiv preprint arXiv:1906.01592*, 2019.

[21] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7278–7286, 2018.

[22] N. Zhao, T.-S. Chua, and G. H. Lee. Few-shot 3d point cloud semantic segmentation. *arXiv preprint arXiv:2006.12052*, 2020.

[23] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.