FEW SHOT LEARNING FOR POINT CLOUD DATA USING MODEL AGNOSTIC META LEARNING

Rishi Puri, Avideh Zakhor, and Raul Puri

Department of EECS, U.C. Berkeley
{puririshi98, avz, raulpuric}@berkeley.edu

ABSTRACT

The ability of deep neural networks to extract complex statistics and learn high level features from vast datasets is proven. Yet current deep learning approaches suffer from poor sample efficiency in stark contrast to human perception. Few shot learning algorithms such as matching networks or Model Agnostic Meta Learning (MAML) mitigate this problem, enabling fast learning with few examples. In this paper, we extend the MAML algorithm to point cloud data using a PointNet Architecture. We construct $N \times K$-shot classification tasks from the ModelNet40 point cloud dataset to show that this method performs classification as well as supervised deep learning methods with the added benefit of being able to adapt after a single gradient step on a single $N \times K$ task. We empirically search for optimal values of N and K for few shot classification and show our method to achieve 90% meta test accuracy compared to traditional PointNet with 89.2%. We also adapt a meta-trained PointNet to a support set of 9, $N = 3, K = 3$, never before seen point clouds which are drawn from an entirely different dataset, ShapeNet. Once adapted the model achieves 7.1/9 classification accuracy on average across 100 query sets of the same classes with new, unique instances. This result far exceeds the supervised Stochastic Gradient Descent (SGD) training result of 3.1/9 accuracy on the query sets which is equivalent to a random baseline.

Index Terms— meta learning, few shot learning, MAML, point clouds, ShapeNet, Modelnet40, 3D

1. INTRODUCTION

Few-shot learning has received a great deal of attention within the machine learning community in recent years. The ability of deep neural networks to extract complex statistics and learn high level features from vast datasets is proven. Yet current deep learning approaches suffer from poor sample efficiency in stark contrast to human perception. A number of few shot learning methods have been proposed to address these issues. They include matching networks [1], which is essentially a differentiable nearest neighbor classifier, prototypical networks [2], which learns prototypical representations, and Model Agnostic Meta Learning (MAML) [3] which learns to fine tune. The de-facto standard for few shot learning is N-shot, K-way classification tasks. As such, the performance of few shot learning algorithms are typically measured by their performance on N-shot, K-way 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 paper, we focus on the problem of few shot learning for point cloud data, using the newly introduced MAML algorithm [3]. The basic idea behind MAML is to learn a network initialisation that can quickly adapt to new tasks — this is a form of meta-learning or learning-to-learn. The end result of this meta-learning is a model that can reach high performance on a new task with as little as a single step of regular gradient descent. The advantage of MAML over matching or prototypical networks is that it works with any differentiable model, i.e. any neural network model that can be back-propagated through.

In this paper, we explore application of MAML to few shot learning of point cloud data. There has been a surge in the number of devices capturing point cloud data, ranging from autonomous driving applications, to consumer devices such as cell phones. Unlike RGB-D data which has a spatial and temporal structure, point cloud data is completely unstructured. Voxelization of point clouds could provide structure to it, but the optimal voxel size might vary from application to application and even within a given point cloud. Too fine of a voxelization results in massive memory and storage requirements, while coarse voxelization could remove features needed for classification and regression. In recent years, PointNet [4] architecture has been proposed as a way to overcome voxelization problems of 3D data by creating feature vectors from point cloud data that are independent of the number of points. In addition, the PointNet architecture was designed with attention to the following three properties: permutation or order invariance, transformation invariance, and point interactions [4]. The PointNet architecture aggregates local and global features to be passed to a multi-layer perceptron. Most importantly, PointNet is a differentiable neural network, and as such meets the requirements of being used

within the MAML framework. In this paper, we apply PointNet within the MAML framework in order to address the few shot learning problem for point clouds. Specifically, we use MAML to train a model for ModelNet40, a dataset comprised of 3D point clouds of 40 classes. We then demonstrate this meta-model to perform well on unseen point clouds drawn from an entirely different dataset, ShapeNet [5], after seeing few examples from that class.

The outline of this paper is as follows in Section 2 we provide an overview of meta learning and PointNet in detail. We describe our method in Section 3. Experimental results are included in Section 4, showing comparable test results to the original PointNet paper with the added benefit of being able to generalize to new data, from ShapeNet, with few examples.

2. BACKGROUND

Meta Learning: Meta learning is a relatively new field with many approaches to solving the same problem: how do we learn to learn? Many state of the art meta learning papers use datasets specifically designed for meta learning such as Omniglot [6] or Meta-Dataset [7] which have a large number of mutually exclusive data classes with limited examples per class. While these data sets provide great benchmarks to compare meta learning algorithms against each other, our work shows MAML's success on a specific kind of task; namely, $N \times K$-shot classification tasks for 3D point clouds of everyday objects.

We choose to use the MAML algorithm [3] to solve few-shot classification, as it is agnostic to model choice, allowing us to use of complex state of the art point cloud processing architectures such as PointNet. MAML is model agnostic since it uses gradient steps to adapt itself, rather than needing architectural modifications to adapt as in many other meta learning methods[3]. This also allows us to perform more straightforward comparisons against traditional supervised deep learning baselines. There are two main classes of meta learning algorithm: those that utilize a specific architecture for meta learning and those that use gradient descent to meta learn agnostic to architecture choice. Older meta learning work tends to fall into the former category and includes Siamese networks [8], Recurrent models [9], MetaPixelCNN [10], Massively Multitask Networks for Drug Discovery [11], Few-Shot Learning With Graph Neural Networks [12], Recommending What Video To Watch Next [13], and BERT [14]. More recent methods tend to follow the latter in that they use gradient descent in an architecture agnostic way. Building upon the foundation of MAML, they include Implicit MAML [15], R2-D2 [16], Alpha MAML [17], CAVIA [18], and MAML++ [19]. These works tend to provide anecdotal improvements without a clear consensus on which is best. In this paper, we choose to use the simplest, successful, model agnostic method [3]. We make this decision the same way a supervised learning work may end up choosing Stochastic Gradient Descent(SGD) or Adam, despite the countless other exotic optimizers that boast anecdotal accuracy gains.

PointNet: PointNet [4] is a novel system for extracting both local and global features from point cloud data. In our work we modify the classification network in PointNet [4] to take $2500\ (x, y, z)$ points as input. We choose 2,500 due to computational constraints. We follow prior work [4] and apply input and feature transformations, and then aggregate point features by max pooling. The input and feature transformations utilize a T-net which learns an affine transformation matrix [4]. The output of the classification network is an N-way classification score, as in $N \times K$-shot. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. We are only using the classification portion of the network in this work, but it would be a simple adjustment to extend our implementation of MAML to tackle segmentation using the same model or object detection using a similar model [4]. PointNet functions within MAML since it can be trained through backpropogation.

3. PROPOSED METHOD

The MAML algorithm for few shot learning is shown in Algorithm 1 [3]. Unlike most deep learning algorithms, MAML does not learn on batches of samples, rather on batches of tasks also known as meta-batches. For each task in a meta-batch we first initialise a new "fast model" using the weights of the base meta-learner. We then compute the gradient and hence a parameter update from samples drawn from that task and update the weights of the fast model i.e. perform typical mini-batch stochastic gradient descent to update the weights of the fast model. This is shown in step 7 of Algorithm 1 with parameter α as the hyperparameter. After the parameter update we sample some more, unseen, samples from the same task and calculate the loss on the task of the updated weights of the "fast model" of the meta-learner; this is shown in step 8 of Algorithm 1. This process is repeated over a number of tasks, as shown in the do loop in Algorithm 1, starting from step 4 and ending in step 9. The final step, i.e. step 10 in Algorithm 1, is to update the weights of the meta-learner by taking the gradient of the sum of losses from the post-update weights from different tasks. This is in fact taking the gradient of a gradient and hence is a second-order update. In this step β is the learning rate hyperparameter of the meta-learner and $p(\mathcal{T})$ is the distribution of tasks. This is the key step as it means we are optimising for the performance of the base model after only a gradient step. The result of this is that the meta-learner can be trained by gradient descent on datasets as small as a single example per class without over-fitting. This step 10 theoretically finds parameters that have learned to incorporate task-specific loss functions and data activations when adapting the parameters to perform well on a new task.

Algorithm 1 MAML for Few-Shot Supervised Learning [3]

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$
11: **end while**

In the formulation presented in Algorithm 1 [3], each task for the inner loop, i.e. steps 4 through 9, contains a query set with N, out of 40 in ModelNet40, classes drawn at random. Each class contains K unique instances of each to compute task-adapted parameters. Here a task is comprised of a $N \times K$ query set and $N \times K$ support set. In Algorithm 1 we refer to the support set as D and the corresponding query set as D'. We deem this few shot or $N \times K$-shot learning. We now establish the terminology used to describe Algorithm 1. The number of tasks sampled for each inner loop from step 4 through 9 is referred to as the meta batch size. To validate, the inner loop beginning with step 4 is run on separate tasks, drawing data from a separate validation set with classes mutually exclusive from the training classes. Then in step 10 of validation, also referred to as "meta-validation", no gradients are taken; rather, accuracy on the query sets is recorded. We also perform a meta-test at the very end of meta-training that follows the same steps of meta-validation on a separate dataset with classes mutually exclusive from both training and validation. We describe this in further detail in Section 4.

ModelNet40 Preprocessing/Data Augmentation: Our data pre-processing system parsed ModelNet40 point cloud data from .off files, a geometry definition file format, to arrays. Any object with less than $2500(x, y, z)$ points is padded with zeros to become $2,500 \times 3$. For objects with more than 2,500 points, we sub-sample 2,500 points of the total uniformly at random. We use random permutation of input points to teach in-variance to permutation as suggested by the original PointNet work [4]. This is, for a point $x = (x_1, .., x_n)$, we feed the PointNet, $f()$, the permuted point $x' = (x_i)_{i \in random\_permutation(1,...,n)}$. Therefore our classification scores $y \in \mathbb{R}^n$, are $y = f(x')$.

Implementation Details: We implemented MAML with the native PyTorch autograd using CUDA10.0 with a NVIDIA 2060 Super GPU for acceleration of training. We implemented PointNet in PyTorch with a classification output layer of N neurons with a soft-max cross entropy loss and an argmax decision rule. We use PyTorch's implementation of Adam for optimization of the meta-parameters. Given limited compute resources for such a high compute task we chose to limit hyperparameter search to a single setting for each N that we found to be successful at converging for all K's.

For $N = 3$ and $N = 5$, we choose meta batch size of 4. The number of finetuning steps for adaptating to new tasks in MAML was limited to 1 with $\alpha = 0.4$. Since validation accuracy is calculated on a single meta-batch, this means that our validation accuracy is measured over $4 \times N \times K$ point clouds. For $N = 3$ we choose $\beta = 4e - 4$ and for $N = 5$ we choose $\beta = 5e - 5$. Meta-training is run over 400,000 meta batches. Meta validation is performed as described before, after each of these training meta batches. We choose a meta-train, meta-validation, meta-test split of 0.6, 0.2, 0.2 for our experiments. Each of meta-train, meta-validation, meta-test contains a mutually exclusive 24, 8, 8 portion of the total 40 classes. We note that this split limits meta-testing to only 8 classes which may seem insufficient. We compensate for this with the additional tests on a held out dataset in Section 4.1.

4. EXPERIMENTAL RESULTS

In this section we describe our experimental results. Section 4.1 includes few shot adaptation of our system to new data and Section 4.2 covers testing and comparison of our system against traditional PointNet and other few shot learning methods.

4.1. Fewshot Adaptation to Unseen Data: We now demonstrate the generalization capability of our trained meta model to a new 3×3-shot classification task. We choose the 3×3-shot meta learner since it performed the best as shown later in Section 4.2. The data is drawn from 3 randomly chosen classes of ShapeNet models [5]. ShapeNet is a dataset with 3D point clouds of everyday objects, similar to ModelNet40. For each class we choose two pairs of three random instances which we preprocessed to have 2,500 (x,y,z) points each, using one pair for the support and one for the query set. We find that adaptation to the support set produces high accuracy of 7/9 on the query set as seen in Figure 1. To compare the performance to supervised methods we perform traditional SGD updates using the same hyperparameters as the original PointNet paper [4] on the support set until accuracy converges and we achieve a random baseline of 3/9 on the query set. We then repeat these experiments while keeping the support set constant and constructing 100 query sets. Each query set contains 3 random instances of each ShapeNet class in the support set. Performing the same procedure as above we achieve an average of $7.1/9$ query set accuracy for our meta learner as seen in Table 1. In contrast, we see $3.1/9$ query set accuracy for our supervised learner. Next, we vary the classes in the query set. For this we create 100 query sets with randomly selected classes. We then measure the accuracy on the corresponding 100 query sets with matching classes and unique instances. This gives us an average query set accuracy of $7.3/9$ for our meta learner and $3.2/9$ for our supervised learner as shown

Query	3×3			3×5			5×3		
	Acc of 9	Acc %	s.d.	Acc of 15	Acc %	s.d.	Acc of 15	Acc %	s.d.
Meta	7.1/9	**78.9%**	3%	11.8/15	78.7%	3%	11.1/15	74.0%	5%
Supervised	3.1/9	34.4%	5%	5.3/15	35.3%	4%	3.2/15	21.3%	7%
Support	3×3			3×5			5×3		
	Acc of 9	Acc %	s.d.	Acc of 15	Acc %	s.d.	Acc of 15	Acc %	s.d.
Meta	7.3/9	**81.1%**	3%	11.4/15	76.0%	7%	10.9/15	72.7%	9%
Supervised	3.2/9	35.6%	6%	5.1/15	34.0%	6%	2.9/15	19.3%	8%

Table 1. *In the Query table we display the averages over different query sets for the same support set. In the Support table we show averages over query sets that match to support sets with differing classes. The first and second sub-columns are fractional and percentage average accuracys. The third is the standard deviation.*

	$N = 3$		$N = 5$	
Accuracy	Val	Test	Val	Test
$K = 1$	91.7%	88.7%	65.0%	68.1%
$K = 2$	87.5%	87.3%	77.5%	76.4%
$K = 3$	91.7%	**90.1%**	87.5%	**88.9%**
$K = 5$	91.7%	89.6%	75.0%	77.8%
$K = 7$	91.7%	88.4%	75.0%	73.2%
Siamese Net	83.7% for 6 classes			
Traditional PointNet	89.2% for 40 classes			

Table 2. *Meta validation and Test for MAML. The meta-test accuracy here is evaluated over 10% of ModelNet40 classes amounting to 1231 instances.*

in Table 1. To expand our results we perform the exact same experiments for 3×5 and 5×3 shot classification with results in Table 1. As seen the meta learner outperforms supervised methods for all settings. We note that the 3x3 learner performs with the highest mean accuracy and lowest standard deviation.

4.2. Meta-Training, Validation, and Testing: We ran $N \times K$ shot MAML on the pre-processed data for $N = \{3, 5\}$, $K = \{1, 2, 3, 5, 7\}$. In order to obtain metrics comparable to the test set accuracy of PointNet on ModelNet40 we perform a meta-test by partitioning our test set into disjoint tasks. The meta-parameters θ are then adapted to support sets of these tasks and evaluated on the query sets. Each query set contains the same classes as its matching support set with different and unique instances.The average few-shot classification accuracy on the query sets is then reported. This produces an accuracy percentage that can be compared fairly to the classification test set accuracy reported in the original PointNet paper [4]. This test is fair as we holdout 20% of data classes for testing. We use half of this, or 10% of the data as a support test set for adaptation. We then evaluate accuracy on the other half, query test sets, which contains 10% of the data classes. This leaves us with an evaluation of classification accuracy on a held out 10% of ModelNet40 classes. This closely follows the testing procedure in PointNet [4]. In Table 2, we see how K affects meta-validation accuracy and meta-test accuracy. We observe that for both $N = 5$ and $N = 3$, meta-test accuracy increases with K until $K = N$, declining after. Specifically, $K = 3$ achieves the best performance for both $N = 3$ and $N = 5$, with the best performance at $N = 3$ and $K = 3$.

As seen in Table 2, $N = 3$ performs better than $N = 5$ in terms of meta-validation accuracy and meta-test accuracy. This may be partially attributed to the fact that $N = 3$ begins training with about 33.3% meta-validation accuracy while $N = 5$ begins with about 20%. These baselines correspond to the results a random guess would achieve. We also note that the highest meta-test results, for 3×3-shot, 3×5-shot, and 3×5-shot learning, slightly outperform the test scores published in the original PointNet paper using supervised learning, tested on 40 classes [4]. Although the comparison is not exact, it is still principled, as both tests involve classifying every unique instance in the test set. We also report meta-test accuracy on 6 classes for Siamese Networks [20] which we note to be lower than supervised results and most results from MAML.

5. CONCLUSIONS AND FUTURE WORK

Our work opens many doors for collaboration with the meta-robotics world. Now that we understand how to effectively train a meta learner to learn from point cloud data, we intend to investigate how a pretrained meta-PointNet can be transferred to function for robotics tasks, e.g. grasping 3D objects. Ideally, a meta-trained robot with a meta-trained 3D vision system could be deployed into the wild with the expectation that it will successfully adapt to new tasks it faces.

(a) (b)

Fig. 1. An example of three classes hat, bottle, guitar being learned with three examples each for: (a) support set; (b) query set. Each picture depicts a point cloud for each instance, with an X indicating the instances with erroneous classification.

6. REFERENCES

[1] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al., "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[2] Jake Snell, Kevin Swersky, and Richard Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, 2017, pp. 4077–4087.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[4] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, 2017, pp. 652–660.

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al., "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[6] Simon Ager, *Omniglot- writing systems and languages of the world*, 2020, https://www.omniglot.com/.

[7] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle, "Meta-dataset: A dataset of datasets for learning to learn from few examples," *arXiv preprint arXiv:1903.03096*, 2019.

[8] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*. Lille, 2015, vol. 2.

[9] Sachin Ravi and Hugo Larochelle, "Optimization as a model for few-shot learning," 2016.

[10] Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas, "Few-shot autoregressive density estimation: Towards learning to learn distributions," *arXiv preprint arXiv:1710.10304*, 2017.

[11] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande, "Massively multitask networks for drug discovery," *arXiv preprint arXiv:1502.02072*, 2015.

[12] Victor Garcia and Joan Bruna, "Few-shot learning with graph neural networks," *arXiv preprint arXiv:1711.04043*, 2017.

[13] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi, "Recommending what video to watch next: a multitask ranking system," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 43–51.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[15] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine, "Meta-learning with implicit gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 113–124.

[16] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi, "Meta-learning with differentiable closed-form solvers," *arXiv preprint arXiv:1805.08136*, 2018.

[17] Harkirat Singh Behl, Atılım Güneş Baydin, and Philip HS Torr, "Alpha maml: Adaptive model-agnostic meta-learning," *arXiv preprint arXiv:1905.07435*, 2019.

[18] Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson, "Fast context adaptation via meta-learning," *arXiv preprint arXiv:1810.03642*, 2018.

[19] Antreas Antoniou, Harrison Edwards, and Amos Storkey, "How to train your maml," *arXiv preprint arXiv:1810.09502*, 2018.

[20] Xuelun Shen, Cheng Wang, Chenglu Wen, Weiquan Liu, Xiaotian Sun, and Jonathan Li, "Discriminative learning of point cloud feature descriptors based on siamese network," in *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2018, pp. 4519–4522.