3D Model Recognition based on two layers Few-Shot Learning Network

Jie Nie[1], Xu Ning[2*], Ming Zhou[3], Ge Yan[3], Zhiqiang Wei[1]

1 College of Information Science and Engineering, Ocean University of China
2 The School of Electrical and Information Engineering, Tianjin University
3 Chengdu Spaceon Measurement & Control Technology Co. Ltd
** Corrsponding Author: Xu Ning.*
Email: ningxu@tju.edu.cn

Abstract

With the development of multimedia technology, 3D model has been applied in many fields such as mechanical design, construction industry, entertainment industry, medical treatment and so on. The number of 3D model is becoming more and more in our lives. Therefore, effective automatic management and classification of 3D models become more and more important. In this paper, we propose a dual-meta-learner model based on LSTM to learn the exact optimization algorithm used to train another two learner neural network classifier in the few-shot regime. The parametrization of our model allows it to learn appropriate parameter updates specifically for the scenario where a set amount of updates will be made, while it can also achieve a general initialization of the learner (classifier) network that allows for quick convergence of training. Our method attains state-of-the-art performance by significant margins.

Keywords: few-shot, meta-learner, 3D model classification

1. Introduction

Deep learning has shown great success [1, 2, 3, 4] in a variety of tasks with large amounts of labeled data in image classification [5, 6] and machine translation [7]. In recent years, success of deep learning methods, in particular, convolutional neural network (CNN) [8], has urged rapid development in various computer vision applications such as image classification in particular CNNs trained on the large datasets such as ImageNet have been shown to learn general purpose image descriptors for a number of vision tasks such as object detection, scene recognition, texture recognition and fine-grained classification. These achievements have relied on the fact that optimization of these deep, high-capacity models requires many iterative updates across many labeled examples. Meanwhile, the recent introduction of depth cameras has made 3D model collection easier. The proliferation of 3D data has promoted the research and interest of automatic classification [9] and retrieval [10, 11, 12, 13] algorithms for 3D models. This is a long-term research task, until recently the introduction of deep learning has achieved satisfactory results. With the increase of 3D model data, the difficulty in obtaining data tags has become a big problem. Applying small sample learning to 3D model classification [14, 15] can solve this problem. 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.

Few-shot classification [16, 17] is a task in which a classifier must be adapted to accommodate new samples which are not seen, but only a few examples of their classes are given in training process. In this setup, instead of a large dataset, we only have a set of datasets in which each class has very few annotated examples. The motivation for this method is that not only adults, but also even children can usually get generalized knowledge after a few examples of a given object. There are many useful applications for models that perform well in this task. First, they help reduce data collection thus we don't need millions of tagged examples to get reasonable performance. Moreover, in many fields, data exhibits features with many different categories but few examples per class. Models that can be promoted from a few examples will be able to capture such data effectively. A naive approach, such as re-training the model on the new data, would severely overfit. 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 [17].

There are two main reasons for gradient-based optimization when faced with some markup examples. First, variants of gradient-based optimization algorithms, such as momentum [18], Adagrad [19], Adadelta [20], and ADAM [21], are not specifically designed for use in a certain number. The performance of the updated constraints is good. Especially when applied to non-convex optimization problems, these algorithms do not have very strong convergence speed guarantees by reasonably selecting hyper-parameters. In addition, they will eventually converge to a good solution after millions of iterations. Second, for each individual data set considered, the network must start with random initialization of its parameters, which greatly impairs its ability to converge to a good solution after several updates. Transfer learning [22, 23] can be applied to mitigate this problem by fine-tuning the pre-training network from another task with more marker data; however, it has been observed that with training networks. The mission deviates from the target mission and the benefits of the pre-training network are greatly reduced [24]. What is needed is a systematic approach to learning a beneficial co-initialization that will serve as a good point for starting the training of the data set under consideration. This will provide the same benefits as transfer learning, but ensuring initialization is the best starting point for fine-tuning.

Previous work has proposed a method to obtain rapid knowledge from a few examples through the concept of meta-learning [25]. Meta-learning suggests building learning problems at the two level. The first is to quickly acquire knowledge in each individual task. This process is guided by a second process, which involves a slower extraction of information learned in all tasks.

Here we present a method to solve the weakness of a neutral network trained by a gradient-based optimization for a few shooting learning problems by constructing problems in a meta-learning environment. We propose an LSTM-based dual meta-learner optimizer that is trained to optimize the learner neural network classifier. Meta-learners capture short-term knowledge in tasks and long-term knowledge common to all tasks. The meta-learners model is trained to quickly aggregate learner classifiers into good resolutions on each task by using the goal of direct capture optimization algorithms with the ability to have good generalization performance given only a certain number of updates. In addition, the development of our meta-learner model allows it to learn the tasks of the learner classifier to co-initialize, which captures the basic knowledge shared between all tasks. We want to generalize a small amount of labeled 3D model data to a large number of unlabeled 3D model data, and then test the classification.

The main contributions of the proposed method are summarized as follows:

- We propose a novel dual-meta-learner model based on two-layer LSTM that uses a small amount of 3d model data to generalize to a large amount of unlabeled 3D model data.

- By generalizing the training on small databases to large databases, the classification effect is comparable to the latest method, but we use less computational cost and time cost.

The rest of the paper 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 new dataset and the related experiments will be shown. We also will discuss the related experimental results in this section; finally, the conclusion will be provided in section 5.

2. RELATED WORK

View-based methods. MVCNN [26] 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 [27] 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, Wang et al. [28] recurrently cluster the views into multiple sets, pool the features in each set and achieve better performance than the original MVCNN.

Pointset-based methods. PointNet proposed by charle et al. [29] directly takes unordered point sets as inputs, addressing the sparsity problem encountered in volume-based methods. Recently, Qi et al. [30] improve PointNet by exploiting local structures induced by the metric space.

A lot of literature on metrics learning [31, 32]; here we summarize the work most relevant to the approach we have proposed. Neighborhood Components Analysis (NCA) [33] learns a Mahalanobis distance to maximize K-nearest-neighbor's (KNN) leave-one-out accuracy in the transformed space. Salakhutdinov and Hinton [34] extend NCA by using a neural network to perform the transformation. Large margin nearest neighbor (LMNN) classification [35] also attempts to optimize KNN accuracy but does so using a hinge loss that encourages the local neighborhood of a point to contain other points with the same label. And the DNet-KNN [36] is another margin-based method that improves upon LMNN by utilizing a neural network to perform the embedding instead of a simple linear transformation. Of these, our method is most similar to the non-linear extension of NCA [34] since we use a neural network to perform the embedding and we optimize a softmax based on Euclidean distances in the transformed space, as opposed to a margin loss. A key difference between our approach and non-linear NCA is that We form softmax directly on the class, rather than a single point calculated from the distance to the prototype representation of each class. This allows each class to have a concise representation that is independent of the number of data points and does not need to store the entire support set for prediction.

Meta-learning has a long history, but it has grown more and more prominent recently, because many people have always advocated it as the key to realizing human wisdom in the future [37]. The ability to learn at two levels (learning in each task while accumulating knowledge about the similarities and differences between tasks) is considered essential for improving AI. Previous work used various techniques in the meta-learning environment. Schmidhuber et al. [38] explores the use of networks that learn how to modify their weights through multiple computational steps on the input. The update of the weight is defined in the form of a parameter that allows the prediction and weight change process to be end-to-end distinguishable. The work of [39] considering learning updates the rules of neural networks that are biologically sound. This property is enforced by allowing the parametric form of the update to only have as input local information at each hidden unit to determine the weight change. Different optimization methods, such as genetic programming or simulated annealing, are used to train the learning rule. This attribute is enforced by allowing the updated parameter form to have a change in weight with only input local information at each hidden unit. Learning rules are trained using different optimization methods, such as genetic programming or simulated annealing.

2.1. Few-shot learning

The best approachs to express few-shot learning is metric learning methods. According to some distance metrics, the deep Siam network training convolutional network embeds the example so that the items in the same class are close to each other and the items in the same category are far away. Matching networks Vinyals et al.[40] refine this idea so that training and testing conditions match, by defining a differentiable nearest neighbor loss involving the cosine similarities of embeddings produced by a convolutional network.

2.2. Shape descriptors

A large number of shape descriptors have been developed for drawing inferences about 3D objects in computer vision and graphical 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 [41] 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. The recent work of Wu et al. [42] is an exception. By learning shape descriptors from the voxel-based representation of an object through 3D convolutional nets, previous 3D shape descriptors were largely ”hand-designed” according to a particular geometric property of the shape surface or volume. For example, we can represent shapes with histograms or bag-of-features models which are constructed out of surface normals and curvatures [43], distances, angles, triangle areas or tetrahedra volumes gathered at randomly sampled surface points [44], properties of spherical functions defined in volumetric grids, local shape diameters measured at densely sampled surface points, heat kernel signatures on polygon meshes, or extensions of the SIFT and SURF feature descriptors to 3D voxel grids [45]. A typical challenge will occur when we develop classifiers and other supervised machine learning algorithms on top of such 3D shape descriptors. That is, the size of organized databases with annotated 3D models is rather limited compared to image datasets, e.g. ModelNet contains about 150K shapes (its 40 category benchmark contains about 4K shapes).

3. Our Approach

In this paper, we propose a novel two layers LSTM for few-shot learning inspired by [46]. We utilize two layers of structure to improve the performance, which can share parameters in the training step. Fig.2 shows the structure of our network. The other reason is that the 3D model can be represented by different modalities such as virtual view and depth image. Here, we hope our approach can also help us to fuse multiple modalities information and bring performance improvement in the 3D shape recognition problem. In the next sections, we will detail our approach.

In the learning setting, We apply the same setting as [46]. We split the dataset D into a set of small batches for training. Each batch includes few-shot training samples, which include the training set D_{train} and the test set D_{test}. However, in our approach, each 3D model represents two modalities such as virtual view and depth view. Here, the virtual view is used to represent the visual information of the 3D model and the depth image is used to represent the structure information of each 3D model. Thus, for each batch, we need to train two times. However, in this paper, we apply the two-steam framework to fuse this two times training. We also share the parameter in the training step. It is the biggest contribution and innovation points.

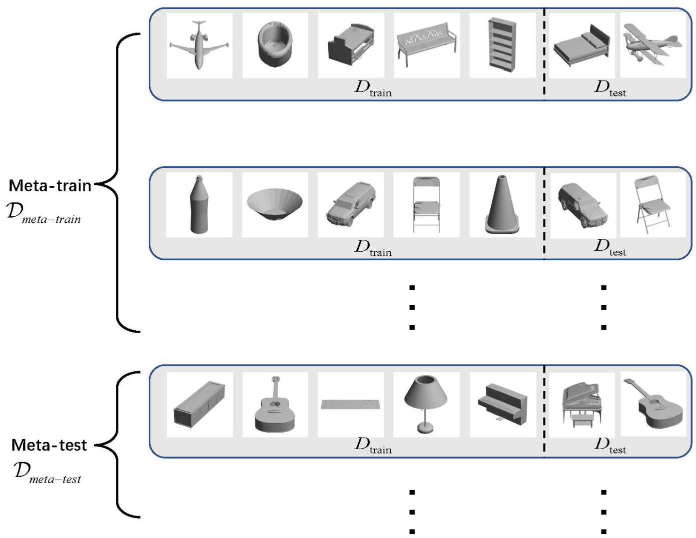

Figure 1: Example of meta-learning setup. The top represents the meta-training set $\mathcal{D}_{meta-train}$, where inside each gray box is a separate dataset that consists of the training set D_{train} (left side) and the D_{test} (right side).

In this paper, our approach mainly includes two parts: 1) the two layers a few-shot learning network; 2) the parameter sharing and preprocessing. We will detail these parts in the next sections.

3.1. two layers Model Structure

First, we need to define the cell state in an LSTM [47]

$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c} \tag{1}$$

if $f_t = 1$, $c_{t-1} = \theta_{\mathrm{t}-1}$ or $\beta_{\mathrm{t}-1}$, $i_t = \alpha_{\mathrm{t}}$ or β_t, and $\widetilde{c} =$ - $\nabla_{\theta_{t-1}}\mathcal{L}_t$ or - $\nabla_{\phi_{t-1}}\mathcal{L}_t$. α_{t} and β_t represent the visual information and structure information respectively. They extracted from the virtual image and depth image.

The goal of this network is to learn an update rule for training a neural network. Here, We utilize the cell of the LSTM to be the parameters of the learner, or $c_t = \theta_\mathrm{t}$ or ϕ_t. Based on the above descriptions, we can define the other parameters of LSTM network like:

$$i_t = \sigma(\mathrm{W}_I \cdot [\nabla_{\theta_{t-1}}\mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}] + \mathrm{b}_I)$$

or

$$i_t = \sigma(\mathrm{W}_I \cdot [\nabla_{\phi_{t-1}}\mathcal{L}_t, \mathcal{L}_t, \phi_{t-1}, i_{t-1}] + \mathrm{b}_I)$$

These two functions are utilized to update the parameters in two networks based on two different modalities training datasets. We also define the forget gate as the following functions.

$$f_t = \sigma(\mathrm{W}_F \cdot [\nabla_{\theta_{t-1}}\mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}] + \mathrm{b}_F)$$

or

$$f_t = \sigma(\mathrm{W}_F \cdot [\nabla_{\phi_{t-1}}\mathcal{L}_t, \mathcal{L}_t, \phi_{t-1}, f_{t-1}] + \mathrm{b}_F)$$

Specially, we need to emphasize that the parameter θ and ϕ are the same parameters in the network. They are shared in the training processing. Here, it is absolutely necessary to describe this process in detail. In each step. we only select one single modality dataset to train the parameter. In the next step, we utilize the other modality to training the other parameter. The training process is alternating. The final target loss function is:

$$\mathcal{L}_{overall} = \mathcal{L}(M_1(X; \theta_{T+1}), Y) + \mathcal{L}(M_2(U; \phi_{T+1}), V). \tag{2}$$

3.2. Parameter Sharing and Preprocessing

In the traditional LSTM model, each training step, the cell state value will be updated. We can easily implement parameter sharing based on the input be a batch of gradient coordinates and loss inputs $(\nabla_{\theta_{t,i}}\mathcal{L}_t,\ \mathcal{L}_t)$ or $(\nabla_{\phi_{t,i}}\mathcal{L}_t,\ \mathcal{L}_t)$. Here, inspired by [48], we normalize the values of the parameter in each training step to guarantee the optimal solution. The normalizing function is followed as:

$$x \rightarrow \begin{cases} (\frac{\log(|x|)}{p}, \mathrm{sgn}(x)) & if|x| \geq e^{-p} \\ (-1, e^p x) & otherwise \end{cases}$$

The key to this paper is the parameter sharing between two different modalities LSTM network. Here, we apply a simple method. By the above description, the training process is alternating. It means that we treat both modality data equally. The update of the cell state and the forget gate is followed by the above equations. We can also say that multi-modalities data are entered alternately and the Parameters update is also alternately.

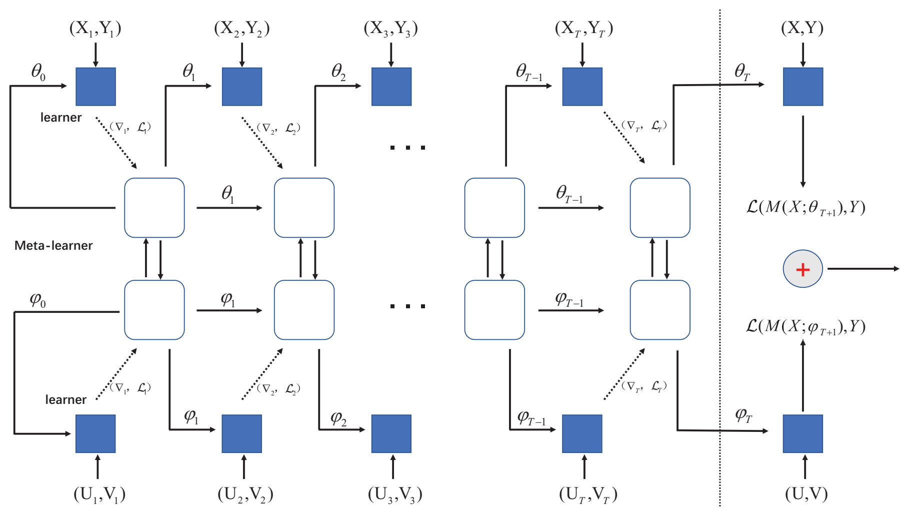

Figure 2: Computational graph for the forward pass of the meta-learner. The dashed line divides examples from the training set D_{train} and test set D_{test}. Every $(X_i,\, Y_i)$ and $(U_i,\, V_i)$ are the i th batch from the different training set whereas (X,Y) and (U,V) are all the elements from the test set. The dashed arrows indicate that we do not back-propagate through that step when training the meta-learner. And the above meta-learner shares the parameters Θ with meta-learner below. We refer to the learner as M_1 and M_2, where $M_1(X;\theta)$ is the output of learner M_1 using parameters ϕ for inputs U and $M_2(U;\phi)$ is the output of learner M_2 using parameters ϕ. We also use ∇t as a shorthand for $\nabla_{\theta_{t-1}}\mathcal{L}_t$ and $\nabla_{\phi_{t-1}}\mathcal{L}_t$

.

4. EXPERIMENT RESULTS

4.1. Datasets

ModelNet We use ModelNet [49] for our training and testing datasets. ModelNet currently contains 127,915 3D CAD models from 662 categories. ModelNet40, a subset including 12,311 models from 40 categories, is well annotated and can be downloaded from the web. The authors also provide a training and testing split on the website, in which there are 9,843 training and 2,468 testing models. We use this train/test split for our experiments. By default, we report classification accuracy on all models in the test set (average instance accuracy). For comparisons with previous work we also report average class accuracy.

ShapeNetCore ShapeNetCore is a subset of the full ShapeNet dataset with single clean 3D models and manually verified category and alignment annotations. It covers 55 common object categories with about 51,300 unique 3D models.

1) We use 20% of ShapeNetCore plus 20% of ModelNet40 data to generalize to the rest 80% of ShapeNetCore data. 2) We use 20% of ShapeNetCore plus 20% of ModelNet40 to generalize to the rest 80% of ModelNet40 data. The learning model is trained through a small portion of large databases, and the experimental results prove that our method is effective and can save computation and time costs.

4.2. The Parameters Selection

We took the value of p at step of 5 and selected the best p-value by the classification accuracy on the ShpeNetCore database. We found that the value

Table 1: The experiments result about selecting parameters p on ShapeNetCore.

value of p	1-shot Classification	5-shot Classification
(1) $p = 1$	$21.51\% \pm 0.42\%$	$23.57\% \pm 0.14\%$
(2) $p = 5$	$33.24\% \pm 0.35\%$	$46.45\% \pm 0.21\%$
(3) $p = 10$	$\mathbf{46.45}\% \pm \mathbf{0.13}\%$	$\mathbf{64.27}\% \pm \mathbf{0.11}\%$
(4) $p = 15$	$30.81\% \pm 0.52\%$	$33.16\% \pm 0.18\%$
(5) $p = 20$	$21.23\% \pm 0.36\%$	$24.86\% \pm 0.48\%$

of $p = 10$ in the above Table worked well in our experiments.

4.3. Comparison with Single Modality

In the paper, we propose a novel two layers few-shot learning model for multiple modalities fusion for 3D model recognition. In order to demonstrate the performance of this approach, we compare with single modality, which means that we only utilize the traditional LSTM structure to leaning the parameter based on the single modality. The corresponding experiments are shown in Table.2.

Table 2: The experiments result about different modalities.

Modality	1-shot Classification	5-shot Classification
(1) Virtual View	41.2% ± 0.24%	59.4% ± 0.25%
(2) Depth Image	42.7% ± 0.16%	61.8% ± 0.15%
(3) Our approach	**46.45%** ± **0.13%**	**64.27%** ± **0.11%**

From the experiment, we can find that Our approach outperforms the results of single modality. It fully shows that our approach effectively fuse multiple modalities information and achieve the better results. We also find that depth modality has the better results than virtual view modality. It also demonstrates that structure information provide more information than virtual information for 3D model representation.

4.4. Compare with state of the art for shape classification

We compare our methods with state of the art for shape classification methods on the ShapeNetCore dataset. Our method is compared against the 3D ShapeNet descriptor by Wu et al.[42], the Spherical Harmonics descriptor (SPH) by Kazhdan et al.[50], the LightField descriptor (LFD) by Chen et al.[51], and Fisher vectors[52] extracted on the same rendered views. PointNet (approach to extract feature based on 3d point cloud) the by Charles et al. [53], GIFT by

Song Bai et al.[27], Multi-view Convolutional Neural Networks (MVCNN) by Hang Su et.al[26].

The experimental results of generalization to ShapeNetCore are summarized in Table 3.In this section, the results of experiments are described, the properties of our model is examined and performance of our method is compared against different approaches. Following [40], the k-shot, N-class classification setting is considered. Under this situation, many related but small training sets of k examples for each of N classes are treated as the input of our method. Firstly, the list of all classes in the data is splited into disjoint sets; Then, they are assigned to each meta-set of meta-training, meta-validation, and meta-testing. In order to generate each instance of a k-shot, N-class task dataset $D = (D_{train}, D_{test}) \in \mathcal{D}$, 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 is produced to ensure that we shall maintain the confidence interval of the mean accuracy to be small.

For the learner,a simple CNN with 4 convolutional layers is used. Each layer is a 3×3 convolution with 32 filters, followed by performing batch normalization, then a ReLU non-linearity, and lastly a 2×2 max-pooling process.Then a final linear layer is followed by a softmax for the number of classes at consideration for our network. The learner assigns the average negative log-probability to the correct class as the loss function $\mathcal{L}$. For the meta-learner, a 2-layer LSTM is used. The first layer is a normal LSTM and the second layer is our modified LSTM meta-learner. At each time step, we compute the learner's loss and gradient on a batch consisting of the entire training set D_{train}, training sets is considered with only a total of 5 or 25 examples. Using a learning rate of 0.001

and with gradient clipping using a value of 0.25, LSTM is trained with ADAM.

1-shot and 5-shot classification are considered for 5 classes in our experiment. 15 examples per class for evaluation is used in each test set.By comparing against recent 3D shape classification techniques, state-of-the-art results are achieved in 3D Recognition. We summary the experimental results of generalization to ShapeNetCore in Table 3 as follows, and we summary the experimental results of generalization to ModelNet40 in Table as follows 4.

Table 3: The experimental results of generalization to ShapeNetCore. Average classification accuracies on ShapeNet with 95% confidence intervals. Marked in bold are the best results for each scenario, as well as other results with an overlapping confidence interval.

Method	#Views	1-shot Classification	5-shot Classification
(1) SPH	-	31.23% ± 0.25%	51.56% ± 0.68%
(2) LFD	-	42.73% ± 0.51%	52.36% ± 0.48%
(3) 3D ShapeNets	-	44.29% ± 0.23%	52.94% ± 0.38%
(4) FV	1	43.56% ± 0.57%	55.56% ± 0.73%
(5) FV, 12 ×	12	43.98% ± 0.36%	55.93% ± 0.21%
(6) CNN	1	45.10% ± 0.29%	56.95% ± 0.68%
(7) CNN, 12 ×	12	46.51% ±0.26%	58.45 % ± 0.54%
(8)PointNet	-	43.34% ± 0.51%	52.86% ± 0.48%
(9)GIFT	12	43.73% ± 0.51%	56.86% ± 0.13%
(10)MVCNN	12	45.10% ±0.29%	56.95% ± 0.68%
(11)Meta-Learner Dual-LSTM (OURS)	12	**46.45**% ± **0.13**%	**64.27**% ± **0.11**%

One of the challenges in 3D shape matching arises from the fact that in many applications, models should be considered to be the same if they only differ by a rotation. Consequently, when comparing two models, a similarity metric implicitly provides the measure of similarity at the optimal alignment. Explicitly solving for the optimal alignment is usually impractical. So, two general methods have been proposed for addressing this issue: (1) Every model is represented using rotation invariant descriptors. (2) Every model is described by a rotation dependent descriptor that is aligned into a canonical coordinate system defined by the model. In this paper, we discuss the limitations of canonical alignment and present a new mathematical tool, based on spherical harmonics, for obtain-

Table 4: The experimental results of generalization to ModelNet40. Average classification accuracy on ModelNet40 with 95% confidence intervals. Marked in bold are the best results for each scenario, as well as other results with an overlapping confidence interval.

Method	#Views	1-shot Classification	5-shot Classification
(1) SPH	-	28.86% ± 0.54%	49.79% ± 0.79%
(2) LFD	-	41.08% ± 0.70%	51.04% ± 0.35%
(3) 3D ShapeNets	-	43.40% ± 0.75%	52.54% ± 0.45%
(4) FV	1	43.13% ±0.37%	55.96% ± 0.23%
(5) FV, 12 ×	12	43.87% ±0.26%	56.73% ± 0.42%
(6) CNN	1	44.13%±0.32%	56.64% ± 0.37%
(7) CNN, 12 ×	12	44.51% ±0.26%	57.25 % ± 0.43%
(8)PointNet	-	44.12% ± 0.65%	52.94% ± 0.45%
(9)GIFT	12	43.25% ± 0.35%	53.40% ± 0.65%
(10)MVCNN	12	44.90% ± 0.53%	58.94% ± 0.43%
(11)Meta-Learner Dual-LSTM (OURS)	12	**46.32**% ± **0.21**%	**62.77**% ± **0.24**%

ing rotation invariant representations. We describe the properties of this tool and show how it can be applied to a number of existing, orientation dependent descriptors to improve their matching performance. The advantage of this is twofold: First, it improves the matching performance of many descriptors. Second, it reduces the dimensions of the descriptor, providing a more compact representation, which in turn makes comparing two models more efficient.

Deep learning has been widely used as a feature extraction technique. Here, we are also interested in how well the features learned from 3D ShapeNets comparing with other state-of-the-art 3D mesh features. We discriminatively fine-tune 3D ShapeNets by replacing the top layer with class labels and use the 5^{th} layer as features. For comparison, we choose Light Field descriptor (LFD) and Spherical Harmonic descriptor (SPH), which performed best among all descriptors[54].

For other algorithms for 3D model classification, our algorithm highlights classification accuracy and saves time costs.

4.5. Implementation Details

In this subsection, we mainly introduce the implementation details of our proposed method. We used two GTX1080TI GPUs to train the Dual-LSTM networks. It takes approximately 6 minutes to train each epoch of the model. SPH, FV ,and LFD are three classic model-based methods for 3D model classification. Thus, they have low computational time in process of model feature extraction. Then, ShapeNets, CNN, PointNet, and MVCNN utilize the popular Deep learning method to structure an effective network to learn 3D model representation function. They need a long time to computation for one 3D model classification. The computational time for all methods is shown in the Table 5.

Table 5: Computational time for all methods.

Methods	Computational Time (One Model)/second	Methods	Computational Time (One Model)/second
SPH	0.1	ShapeNets	2
LFD	0.15	CNN	0.24
FV	0.11	CNN-12	3.1
FV-12	0.9	PointNet	3.15
GIFT	3.5	MVCNN	2.4
Our Approach	1.35		

5. Conclusion

We described an Dual-LSTM-based model for meta-learning, which is inspired by the parameter updates suggested by gradient descent optimization algorithms. Our Dual-LSTM meta-learner uses its state to represent the learning updates of the parameters of a classifier. It is trained to discover both a good initialization for the learner's parameters and a successful mechanism for updating the learner's parameters to a given small training set for some new classification task. Our experiments demonstrate that our approach outperforms natural baselines and is competitive to the state-of-the-art in metric learning for few-shot learning. In this work, we focus our study to the few-shot and few-classes setting. However, it would be more valuable to train meta-learners that can perform well across a full spectrum of settings, i.e. for few or lots of training examples and for few or lots of possible classes. Thus we consider to move our future towards this more challenging scenario. Through the experimental results, we can conclude that our method can guarantee the validity, while greatly reducing the computational cost and time cost, which is meaningful for the application.

References

References

[1] J. Ou, Y. Li, Vector-kernel convolutional neural networks, Neurocomputing 330 (2019) 253–258.

[2] S. P. Adhikari, C. Yang, K. Slot, M. Strzelecki, H. Kim, Hybrid no-propagation learning for multilayer neural networks, Neurocomputing 321 (2018) 28–35.

[3] J. Yu, Z. Kuang, B. Zhang, W. Zhang, D. Lin, J. Fan, Leveraging content sensitiveness and user trustworthiness to recommend fine-grained privacy settings for social image sharing, IEEE Transactions on Information Forensics and Security 13 (5) (2018) 1317–1332.

[4] C. Hong, J. Yu, J. Wan, D. Tao, M. Wang, Multimodal deep autoencoder for human pose recovery, IEEE Transactions on Image Processing 24 (12) (2015) 5659–5670.

[5] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition (2015) 770–778.

[6] J. Yu, X. Yang, F. Gao, D. Tao, Deep multimodal distance metric learning using click constraints for image ranking, IEEE Transactions on Cybernetics PP (99) (2016) 1–11.

[7] A. V. D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio.

[8] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: International Conference on Neural Information Processing Systems, 2012.

[9] X. Chen, Y. Chen, K. Gupta, J. Zhou, H. Najjaran, Slicenet: A proficient model for real-time 3d shape-based recognition, Neurocomputing 316 (2018) 144–155.

[10] J. Yu, R. Yong, C. Bo, Exploiting click constraints and multi-view features for image re-ranking, IEEE Transactions on Multimedia 16 (1) (2013) 159–168.

[11] J. Yu, D. Tao, M. Wang, Y. Rui, Learning to rank using user clicks and visual features for image retrieval, IEEE Transactions on Cybernetics 45 (4) (2015) 767–779.

[12] J. Zhang, J. Yu, D. Tao, Local deep-feature alignment for unsupervised dimension reduction, IEEE Trans Image Process.

[13] J. Yang, J. Zhao, Q. Sun, 3d model retrieval using constructive-learning for cross-model correlation, Neurocomputing 275 (2018) 1–9.

[14] H. Guo, J. Wang, Y. Gao, J. Li, H. Lu, Multi-view 3d object retrieval with deep embedding network, IEEE Transactions on Image Processing 25 (12) (2016) 5526–5537.

[15] K. Lu, R. Ji, J. Tang, Y. Gao, Learning-based bipartite graph matching for view-based 3d model retrieval, IEEE Transactions on Image Processing 23 (10) (2014) 4553–4563.

[16] E. G. Miller, N. E. Matsakis, P. A. Viola, Learning from one example through shared densities on transforms, in: Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, 2000, pp. 464–471 vol.1.

[17] B. M. Lake, R. Salakhutdinov, J. Gross, J. B. Tenenbaum, One shot learning of simple visual concepts, Cognitive Science 33 (33).

[18] B. Y. Nesterov, A method of solving a convex programming problem with convergence rate o(1/, in: 1 k2, 2010.

[19] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research 12 (7) (2011) 257–269.

[20] M. D. Zeiler, Adadelta: An adaptive learning rate method, arXiv: Learning.

[21] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, international conference on learning representations.

[22] R. Caruana, Learning many related tasks at the same time with backpropagation (1994) 657–664.

[23] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: A deep convolutional activation feature for generic visual recognition, international conference on machine learning (2014) 647–655.

[24] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks, neural information processing systems (2014) 3320–3328.

[25] S. Thrun, Lifelong Learning Algorithms, Kluwer Academic Publishers, 1998.

[26] H. Su, S. Maji, E. Kalogerakis, E. G. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in: Proc. ICCV, 2015.

[27] B. Song, B. Xiang, Z. Zhou, Z. Zhang, L. J. Latecki, Gift: A real-time and scalable 3d shape search engine, in: Computer Vision & Pattern Recognition, 2016.

[28] C. Wang, M. Pelillo, K. Siddiqi, Dominant set clustering and pooling for multi-view 3d object recognition.

[29] R. Q. Charles, S. Hao, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: IEEE Conference on Computer Vision & Pattern Recognition, 2017.

[30] C. R. Qi, Y. Li, S. Hao, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space.

[31] A. Bellet, A. Habrard, M. Sebban, A survey on metric learning for feature vectors and structured data, Computer Science.

[32] B. Kulis, Metric learning:a survey, Foundations & Trends® in Machine Learning 5 (4).

[33] J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, Neighbourhood components analysis, in: International Conference on Neural Information Processing Systems, 2004, pp. 513–520.

[34] R. Salakhutdinov, G. E. Hinton, Learning a nonlinear embedding by preserving class neighbourhood structure (2007) 412–419.

[35] K. Q. Weinberger, L. K. Saul, Distance metric learning for large margin nearest neighbor classification, Journal of Machine Learning Research 10 (2009) 207–244.

[36] R. Min, D. A. Stanley, Z. Yuan, A. J. Bonner, Z. Zhang, A deep non-linear feature mapping for large-margin knn classification (2009) 357–366.

[37] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, S. J. Gershman, Building machines that learn and think like people, Behavioral & Brain Sciences 40 (2016) 1.

[38] J. Schmidhuber, Learning to control fast-weight memories: An alternative to dynamic recurrent networks., Neural Computation 4 (1) (1992) 131–139.

[39] Y. Bengio, S. Bengio, J. Cloutier, Learning a synaptic learning rule, in: Ijcnn-91-Seattle International Joint Conference on Neural Networks, 2002, p. 969 vol.2.

[40] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, D. Wierstra, Matching networks for one shot learning.

[41] C. Wang, M. Cheng, F. Sohel, M. Bennamoun, J. Li, Normalnet: a voxel-based cnn for 3d object classification and retrieval, Neurocomputing.

[42] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes, computer vision and pattern recognition (2015) 1912–1920.

[43] B. K. P. Horn, Extended gaussian images, Proceedings of the IEEE 72 (12) (1984) 1671–1686.

[44] S. Chaudhuri, V. Koltun, Data-driven suggestions for creativity support in 3d modeling, in: ACM SIGGRAPH Asia, 2010, p. 183.

[45] J. Knopp, M. Prasad, G. Willems, R. Timofte, L. Van Gool, Hough transform and 3d surf for robust three dimensional classification, european conference on computer vision 6316 (2010) 589–602.

[46] S. Ravi, H. Larochelle, Optimization as a model for few-shot learning.

[47] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.

[48] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, N. De Freitas, Learning to learn by gradient descent by gradient descent.

[49] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes (2014) 1912–1920.

[50] M. Kazhdan, T. Funkhouser, S. Rusinkiewicz, Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors, in: L. Kobbelt, P. Schroeder, H. Hoppe (Eds.), Eurographics Symposium on Geometry Processing, The Eurographics Association, 2003. doi:10.2312/SGP/SGP03/156-165.

[51] D. Chen, X. Tian, Y. Shen, M. Ouhyoung, On visual similarity based 3d model retrieval, Computer Graphics Forum 22 (3) (2003) 223–232.

[52] F. Perronnin, C. Dance, Fisher kernels on visual vocabularies for image categorization, in: IEEE Conference on Computer Vision & Pattern Recognition, 2008.

[53] R. Q. Charles, H. Su, M. Kaichun, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, computer vision and pattern recognition (2017) 77–85.

[54] D. Y. Chen, X. P. Tian, Y. T. Shen, O. Ming, On visual similarity based 3d model retrieval, Computer Graphics Forum 22 (3) (2010) 223–232.