Abstract

3D data analysis becomes a quite prevalent topic these years because of its numerous applications, including autonomous vehicles, robotics and so on. In contrast to 2D data such as images which are represented by 2D grids of pixels, there are several different data representations for 3D data. For instance, contours and skeletons of 3D models are usually detailedly depicted by thousands of tiny triangle meshes on their surfaces, and RGBD (Red, Green, Blue and Depth) images that includes depth information describing spatial structures to some extend are frequently used to portray indoor scenes. Compared with RGBD images, mesh-based representaion captures 3D structures more detailedly and specifically, which is the reason why meshes are commonly used in computer graphics. However, mesh-based data is incompatible with recent popular data-driven machine learning algorithms. Therefore, some types of preprocessing are propsed to convert mesh-based data to other compatible representations.

The first and the most straightforward preprocessing is similar to the concept of rendering in computer graphics. Each model is rendered into a batch of RGB images from different view points, with each image capturing a portion of 3D structure, and the entire set of images forms a comprehensive description. Afterwards, any algorithm that is capable of consuming images can be leveraged, whether it is a data-drived one or a hand-crafted one. The second preprocessing is intuitive. It imitates 2D images so that it trys to segment mesh-based models into 3D grids of voxels. Each voxel may contain a variety of features such as spatial information, information of surface direction, color information and so on. Thus, algorithms processing 2D grids of pixels can be generalized to 3D grids of voxels. The third prepocessing is quite convenient and practical for real world usage. It samples thousands of points from meshes so that every model or indoor scene is represented by a cloud of points, and points may contain rich information like 3D pixels. This prepocessing is quite convenient in reality because the generation of point clouds can get rid of sampling from meshes, but obtained by 3D LIDAR instead in industry.

From the above-mentioned three prepocessing techniques, it is apparent that each technique has its own advantages and drawbacks. The rendering technique can take advantage of recent powerful Convolutional Neural Networks (CNNs) directly. And the only trivial revision is to design a strategy synthesizing features of a set of images from different view points. Nevertheless, rendering technique is related to loss of 3D information. Much spatial information is discarded during rendering, and one compensation is to increase view points, which leads to much more computational cost on the other hand. The second voxelization technique can capture 3D spatial information quite well. However, similar to images, contours of models become blur when resolution is low. Therefore, in order to generate high-resolution voxelized models, the size of voxels must be tiny so that more voxels are involved in every dimension. It is obvious computational cost explodes cubically with respect to resolution. The last point clouds technique can also depict 3D spatial information quite well, and importantly, it is computationally efficient. In order to obtain more detailed 3D structure, more points should be sampled which causes linearly incremental computational

cost. Nonetheless, the problematic aspect of point cloud technique is that sparsity and orderlessness are inborn attributes of point clouds, which means recent state-of-the-art algorithms that often take dense and ordered grids as input are no longer suitable. So, proposing novel algorithms that are capable of processing sparse and orderless data is the core of my work.

Two novel algorithms are proposed. One is Extreme Learning Machine (ELM) based algorithm, and the other is the generalization of CNNs to point clouds using Bilinear Pooling.

ELM is an effective and efficient Single Layer Feed-forward Neural-network (SLFN) that is capable of doing diverse tasks, e.g., regression, classification, clustering, etc. It involves no iterative update schemes like gradient descent during training procedure, but it randomly generates input weights and bias and minimizes the training error and the norm of output weights simultaneously instead. According to ELM theory, the smaller norm of weights feed-forward neural network has when it minimizes the training error, the better generalization perfromance it reaches. Therefore, ELM is considered to be an universal approximator theoretically. Extreme Learning Machine Auto-Encoder (ELM-AE) is a variant of ELM whose output is exactly the input of the network. Similar to other auto-encoders, ELM-AE is unsupervised and tries to learn feature representations easy to reconstruct the input, which is a significant attribute of meaningful representations. However, ELM-AE is critically distinct from deep auto-encoders in the aspect that it gets rid of gradient-based update schemes and utilizes several matrix operations to determine the weights instead. Hence, ELM-AE is considered way faster than deep auto-encoders, especially for training, and it can also learn meaningful representations according to ELM theory. Moreover, ELM-AEs can be stacked one after another to boost the learning capability. So, in the first algorithm, a stack of ELM-AEs is used as a feature extractor of point clouds whose output is fed into a sigle-layer ELM classifier. I term this algorithm as Point Clouds-based Extreme Learning Machine Autoencoder (PC-ELM-AE). PC-ELM-AE achieves decent performance on two benchmarks, ModelNet10 and ModelNet40.

Bilinear Pooling or Bilinear Models are proposed to tackle fine grained recognition task. Its variants, e.g., Compact Bilinear Model and Factorized Low Rank Bilinear Pooling, realize impressive performance on other diverse tasks like semantic segmentation and Natural Language Processing (NLP), which demonstrates the effectiveness of Bilinear Pooling. Bilinear Pooling sums over a set of matrices generated by outer product of features on the same location. The corresponding bilinear-pooled feature abandons original order information, which coincides with the demand of point clouds processing. Consequently, I incorporate Bilinear Pooling method into every local neighborhood to generate an order-irrelevant feature that captures second order statistics. Afterwards, 2D convolutional layers are applied to the corresponding order-irrelevant feature of each local neighborhood to extract higher level representation. However, one problem of Bilinear Pooling is that bilinear-pooled features are often of extremely high dimension. On one hand, extremely high dimensional features are composed of a lot of redundant components which leads to excessive storage cost and computational cost. On the other hand, more parameters are needed in convolutional layers to process high dimensional features, resulting in overfitting easily. Hence, I leverage Factorized Low Rank Bilinear Pooling instead of the tranditional one. Factorized Low Rank Bilinear Pooling explic-

itly constrains weight matrices to be low-rank, which tremendously reduce the number of parameters, and explicitly computing bilinear-pooled features is not required, which immensely save storage cost and computational cost. I evaluate the proposed Factorized Low Rank Bilinear Pooling method on two classification benchmarks, ModelNet10 and ModelNet40, and one part segmentation benchmark, ShapeNetPart. It realizes state-of-the-art performance not only on classification datasets but also on segmentation dataset.