3DmFV: 3D Point Cloud Classification in Real-Time using Convolutional Neural Networks

Yizhak Ben-Shabat[1], Michael Lindenbaum[2], Anath Fischer[3]

***Abstract*—Modern robotic systems are often equipped with a direct 3D data acquisition device, e.g. LiDAR, which provides a rich 3D point cloud representation of the surroundings. This representation is commonly used for obstacle avoidance and mapping. Here, we propose a new approach for using point clouds for another critical robotic capability, semantic understanding of the environment (i.e. object classification). Convolutional neural networks (CNN), that perform extremely well for object classification in 2D images, are not easily extendible to 3D point clouds analysis. It is not straightforward due to point clouds' irregular format and a varying number of points. The common solution of transforming the point cloud data into a 3D voxel grid needs to address severe accuracy vs memory size tradeoffs. In this paper we propose a novel, intuitively interpretable, 3D point cloud representation called 3D Modified Fisher Vectors (*3DmFV*). Our representation is hybrid as it combines a coarse discrete grid structure with continuous generalized Fisher vectors. Using the grid enables us to design a new CNN architecture for real-time point cloud classification. In a series of performance analysis experiments, we demonstrate competitive results or even better than state-of-the-art on challenging benchmark datasets while maintaining robustness to various data corruptions.**

***Index Terms*—Deep Learning in Robotics and Automation, Recognition, Computer Vision for Transportation, Computer Vision for Other Robotic Applications**

I. Introduction

OBJECT classification is an important skill for autonomous robots operating in unknown environments. Direct acquisition of 3D geometrical data using, e.g., LiDAR and RGBD cameras, providing point clouds, has become a common choice for autonomouse vehicles such as cars [1], and quadrotors [2]. These sensors are commonly used for obstacle avoidance and mapping but have great potential to contribute also to a semantic understanding of the environment.

We propose a new approach for analyzing a 3D point cloud, using deep neural networks, and especially convolutional neural networks (ConvNets). ConvNets have shown remarkable performance in image analysis. Adapting them to point clouds is not, however, straightforward. ConvNets are built for input data arranged in fixed size arrays, on which linear space invariant filters (convolutions) may be applied. Point clouds, unfortunately, are unstructured, unordered, and contain a varying number of points. Therefore, they do not fit naturally into a spatial array (grid).

Several methods for extending ConvNets to 3D point cloud analysis have been proposed [3], [4], [5]. A common approach is to rasterize the 3D point data into a 3D voxel grid (array), on which ConvNets can be easily applied. This approach, however, suffers from a tradeoff between its computational cost and approximation accuracy. We discuss this approach, as well as other common point cloud representations, in Section II.

We take a different approach and use a new point cloud hybrid representation, the 3D Modified Fisher Vector (*3DmFV*). The representation describes points by their deviation from a Gaussian Mixture Model (GMM). It has some similarity to the Fisher Vector representation (FV) [6], [7] but modifies and generalizes it in two important ways: the proposed GMM is specified using a set of uniform Gaussians with centers on a 3D grid, and the components characterizing the set of points, that, for Fisher vectors, are *averages* over this set, are generalized to other functions of this set.

The representation is denoted hybrid because it combines the discrete structure of a grid with the continuous nature of the components. It has several advantages. First, because it keeps the continuous properties of the point cloud, it retains some of the point set's fine detail and, under certain conditions, is lossless, invertible, and is therefore equivalent to featureless input. Second, the grid-like structure makes it possible to use ConvNets, which yields excellent classification accuracy even with low resolutions (e.g. $8\times8\times8$). Finally, each component of the proposed representation is intuitively interpretable.

The main contribution of this work is a new, accurate, robust, and real-time object classification method for 3D point clouds. Additional contributions are:

- A new hybrid representation for 3D point clouds (3DmFV) which is structured and order independent.
- A new deep ConvNet architecture design (3DmFV-Net) for point cloud classification, obtaining state of the art results on CAD and LiDAR data.
- A thorough empirical analysis is conducted on the stability of our method.

We first review related work on 3D classification and the FV representation in Section II. Then, in Section III we introduce and discuss the 3DmFV representation and 3DmFV-Net architecture. The classification results are presented in Section IV. Finally, Section V concludes the paper and adds some new insight.

Manuscript received: February 24, 2018; Revised: May 24, 2018; Accepted: June 6, 2018.

This paper was recommended for publication by Jana Kosecka upon evaluation of the Associate Editor and Reviewers' comments. This Work was supported by the Ministry of Economy and Industry and by Israel Science Foundation.

[1]Yizhak Ben-Shabat and Anath Fischer are with the Department of Mechanical Engineering, Techion IIT, Haifa, Israel, sitzikbs@gmail.com

[2]Michael Lindenbaum is with the Department of Computer Science, Techion IIT, Haifa, Israel, mic@cs.technion.ac.il

Digital Object Identifier (DOI): see top of this page.

II. Related Work

A. Deep learning on 3D data

Point cloud features - Handcrafted features for point clouds have shown adequate performance for many tasks. They can be divided into two main groups: local descriptors [8], [9], [10], [11] and global descriptors [12], [13], [14]. Some descriptors were specifically designed for the task of object classification in outdoor LiDAR data [15], [16]. While the comprehensive performance evaluation in [17] suggests guidelines for feature selection, it remains non-trivial and highly data specific.

Deep learning on 3D representations - 3D data is commonly represented using one of the following representations: (a) Multi-view, (b) Volumetric grid, (c) Mesh, (d) Point clouds. Each representation requires a different approach for modifying the data to the form required by deep learning methods.

Rendering 2D images of a 3D object from multiple views, as in [5], transforms the learning domain from 3D to the well-researched 2D domain. Information is lost in the projection process, but using multiple projections partially compensates.

The volumetric, voxelized, representation discretizes 3D space similarly to an image discretizing a camera projection plane. This enables a straightforward extension of learning using 3D CNNs [3], [4], [5]. A recent improvement applies an ensemble of very deep networks that extend the principles of Inception [18] and Resnet [19] to voxelized data [20], thus achieving the highest accuracy to date at the price of high computational cost and training time of weeks.

A volumetric representation is associated with a quantization tradeoff: choosing a coarse grid leads to quantization artifacts and to substantial loss of information, whereas choosing a fine grid significantly increases the number of voxels, which are mostly empty but still induce a high computational cost. The computational cost may be reduced by using efficient data structures. For example, OctNet [21] uses unbalanced Octrees as a space partitioning function within the network architecture. Similarly, O-CNN [22] uses an Octree variation that is optimized for performing convolutions on the GPU. Alternatively, computational improvement may be achieved by specifying efficient convolution operators, such as sparsity aware convolutions in Vote3Deep [23], hash map based sparse convolutions [24], or attenion based sparse convolutions (ILA-SCNN) [25].

For the mesh representation, Spectral ConvNets [26], [27], [28] and anisotropic ConvNets [29] can be applied. These approaches utilize mesh topological structure, which is not always available.

The point cloud representation is challenging because it is both unstructured and point-wise unordered. To overcome these challenges, the PointNet approach [30], [31] applies a symmetric function that is insensitive to the order, on a high-dimensional representation of the individual points. The Kd-Network [32] imposes a kd-tree structure on the points and uses it to learn shared weights for nodes in the tree.

Representing point clouds using Gaussians - Gaussians were used in several previous works to represent point clouds. The Normal Distribution Transform (NDT) [33] is a piecewise continuous representation which models the distribution of points by a collection of local normal distributions on several overlapping grid cells. It quantifies the probability to find a point in a given position with respect to the cell that contain the point. Another approach represents each point by three saliency features (scatter, linear, and surface) [34] which are linear combinations of the eigenvalues of the local point sets' covariance matrix. The saliency features distribution is then learned by fitting a GMM using the Expectation Maximization (EM) algorithm.

Here, we propose the 3DMFV representation for point clouds that can be used efficiently as input to a CNN. Although there is some similarity to NDT, our 3DmFV representation includes derivatives of all points w.r.t the Gaussians parameters, rendering it continuous rather than piecewise-continuousness.

B. Fisher vectors

Before the age of deep learning, the bag of visual words (BoV) [35] was a popular choice for image classification tasks. It extracted a set of local descriptors and assigned each of them to the closest entry in a codebook (visual vocabulary), leading to a histogram of occurrences. Perronin and Dance [6], and Perronnin and Thomas [36] proposed an alternative descriptor aggregation method, called the Fisher Vector (FV), based on the Fisher Kernel (FK) principle of [37]. The FV characterizes data samples of varying sizes by their deviation from a generative model, in this case a Gaussian Mixture Model (GMM). It does so by computing the gradients of the sample's log-likelihood w.r.t. the model parameters (i.e., weight, mean and covariance). FV can be viewed as a generalization of the BoV as the histogram is closely related to the derivative w.r.t. the weight. Furthermore, it was shown in [37] that, when the label is included as a latent variable of the generative model, the FK is asymptotically as good as the maximum a posteriori (MAP) decision rule for this model. The FV representation is optimal and sample size independent, this makes it a natural choice for representing point cloud data.

In the context of image classification, the combination of FVs and DNNs was already considered [38], [39]. A network composed of Fisher layers was suggested in [38]. Each layer performs semi-local FV encoding (on dense handcrafted features) followed by a dimensionality reduction, spatial stacking, and normalization. A network composed of unsupervised and supervised layers was proposed in [39]. The unsupervised layers calculate features, FVs, and reduce their dimension. They are followed by supervised fully connected layers, trained with back propagation.

The proposed 3DmFV shares some properties with the representation types above. Like the volumetric approach, it is based on a grid, but not a grid of voxels. It thus maintains the grid structure, which makes it a convenient input to a ConvNet, but it suffers less from quantization. Like the PointNet approach, its features are symmetric functions, making them order and structure independent. The architecture we propose, like that of [39], combines unsupervised and supervised layers. However, it relies on the spatial properties of point clouds, which enables the use of ConvNets, substantially improving its performance.

III. THE 3DMFV-NET

The proposed 3DmFV-Net classification architecture consists of two main modules. The first converts an input point cloud to the 3D modified Fisher vector (3DmFV) representation and the second processes it in a CNN architecture. These main modules are illustrated in Figure 1 and described below.

A. Describing point clouds with Fisher vectors

The proposed representation builds on the well-known Fisher vector representation. We start by formally describing the Fisher vectors (following the formulations and notation of [7]) in the context of 3D points, and then discuss some of their properties that lead to the proposed generalization and make them attractive as input to deep networks.

The Fisher vector is based on the likelihood of a set of vectors associated with a Gaussian Mixture model (GMM). Let $X = \{\boldsymbol{p}_t \in \mathbb{R}^3, t = 1, ...T\}$ be the set of 3D points (the point cloud), where T denotes the number of points in the set. Let λ be the set of parameters of a K component GMM $\lambda = \{(w_k, \mu_k, \Sigma_k), k = 1, ...K\}$, where w_k, μ_k, Σ_k are the mixture weight, expected value, and covariance matrix of k-th Gaussian. The likelihood of a single 3D point (or vector) p associated with the k-th Gaussian density is

$$u_k(\boldsymbol{p}) = \frac{1}{(2\pi)^{3/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{p}-\mu_k)'\Sigma_k^{-1}(\boldsymbol{p}-\mu_k)\right\}. \tag{1}$$

The likelihood of a single point associated with the GMM density is therefore:

$$u_\lambda(\boldsymbol{p}) = \sum_{k=1}^{K} w_k u_k(\boldsymbol{p}). \tag{2}$$

Given a specific GMM, and under the common independence assumption, the Fisher vector, $\mathscr{G}_\lambda^X$, may be written as the sum of normalized gradient statistics, computed here for each point $\boldsymbol{p}_t$:

$$\mathscr{G}_\lambda^X = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(\boldsymbol{p}_t), \tag{3}$$

where L_λ is the square root of the inverse Fisher Information Matrix. The following change of variables, from $\{w_k\}$ to $\{\alpha_k\}$, ensures that $u_\lambda(x)$ is a valid distribution and simplifies the gradient calculation [40]:

$$w_k = \frac{exp(\alpha_k)}{\sum_{j=1}^{K} exp(\alpha_j)}. \tag{4}$$

The soft assignment of point $\boldsymbol{p}_t$ to Gaussian k is given by:

$$\gamma_t(k) = \frac{w_k u_k(\boldsymbol{p}_t)}{\sum_{j=1}^{K} w_j u_j(\boldsymbol{p}_t)}. \tag{5}$$

The normalized gradient components may be written explicitly as:

$$\mathscr{G}_{\alpha_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k), \tag{6}$$

$$\mathscr{G}_{\mu_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{\boldsymbol{p}_t - \mu_k}{\sigma_k}\right), \tag{7}$$

$$\mathscr{G}_{\sigma_k}^X = \frac{1}{\sqrt{2w_k}} \sum_{t=1}^{T} \gamma_t(k) \left[\frac{(\boldsymbol{p}_t - \mu_k)^2}{\sigma_k^2} - 1\right]. \tag{8}$$

These expressions include the normalization by L_λ under the assumption that $\gamma_t(k)$ (the point assignment distribution) is approximately sharply peaked [7]. For grid uniformity reasons, discussed in Section III-D, we adopt the common practice and work with diagonal covariance matrices. The Fisher vector is formed by concatenating all of these components:

$$\mathscr{G}_{FV_\lambda}^X = \left(\mathscr{G}_{\alpha_1}^X, ..., \mathscr{G}_{\alpha_k}^X, \mathscr{G}_{\mu_1}^{X\prime}, ..., \mathscr{G}_{\mu_k}^{X\prime}, \mathscr{G}_{\sigma_1}^{X\prime}, ..., \mathscr{G}_{\sigma_k}^{X\prime}\right). \tag{9}$$

To avoid the dependence on the number of points, the resulting FV is normalized also by the sample size T [7]:

$$\mathscr{G}_{FV_\lambda}^X \leftarrow \frac{1}{T}\mathscr{G}_{FV_\lambda}^X. \tag{10}$$

See [7] for derivations, efficient implementation, and more details.

B. Advantages of Fisher vectors as inputs to DNNs

A Fisher vector, representing a point set, may be used as an input to a DNN. It is a fixed size representation of a possibly variable number of points in the cloud. Its components are normalized sums of functions of the individual points. Therefore, FV representation of a point set is invariant to order, structure, and sample size.

Using a vector of non-learned features instead of the raw data goes against the common wisdom of deep neural network users since learned features, obtained by end to end training, are supposed to be optimal. Non-learned features choose specific data properties and may lose important characteristics. This is true especially when the feature extraction involves discretization, as is the case with voxel based representation. We argue, however, that the Fisher vector representation, being continuous on the point set, suffers less from this disadvantage. We shall give three arguments in favor of this claim.

a. Equation counting argument - Consider a K component GMM. A set of T points, characterized by $3T$ scalar coordinates, is represented using $7K$ components of the Fisher vector, every one of which is a continuous function of the $3T$ variables. Can we find another point set associated with the same Fisher components? We argue that for $T < 7K/3$, the set of equations specifying unknown points from known Fisher components is over-determined and that it is likely that the only solution, up to point permutation, is the original point set. While this equation counting argument is not a rigorous proof, such claims are common for sets of polynomial equations and for points in general position. If the solution is indeed unique, then the Fisher representation is lossless and using it is equivalent to using the raw point set itself.

b. Reconstructing the represented point structure in simplified, isolated cases

b.1 A single Gaussian representing a single point - Here, $T = 1$. By the sharply peaked $\gamma_t(k)$ assumption, there is

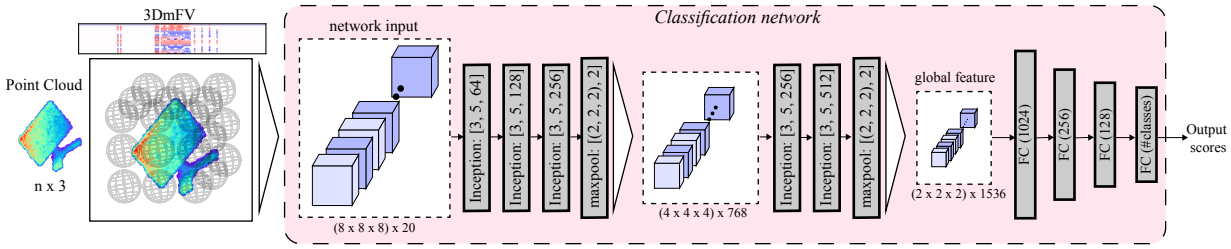

Fig. 1. 3DmFV-Net architecture

only one Gaussian for which $\gamma_t(k) = 1$. Inserting its FV components in Eq. 7 provides the point location:

$$\boldsymbol{p}_1 = \sigma_k \mathscr{G}^X_{\mu_k} + \mu_k. \tag{11}$$

b.2 A single Gaussian representing multiple points on one plane - We now show that, given a set of points sampled on a plane, it is possible to reconstruct a plane from the FV representing the points. The plane equation is $\hat{n}^T p = \rho$, where $\hat{n} = (a, b, c)^T$ is the unit normal to the plane and ρ is its distance from the origin. Using the assumption that $\gamma_t(k)$ is approximately sharply peaked [7], we consider the k-th Gaussian and the T points for which $\gamma_t(k) \approx 1$. For this Gaussian, eq. 7 is simplified to:

$$\mathscr{G}^X_\mu = \frac{1}{\sigma\sqrt{w}} \sum_{t=1}^{T} (\boldsymbol{p}_t - \mu). \tag{12}$$

Changing the coordinate system to $x'y'z'$, for which the origin is at the Gaussian center and an axis x' coincides with $\hat{n}$, leads to the following expression for the plane parameters:

$$a = \frac{\mathscr{G}_{\mu_x}}{\|\mathscr{G}_\mu\|}, b = \frac{\mathscr{G}_{\mu_y}}{\|\mathscr{G}_\mu\|}, c = \frac{\mathscr{G}_{\mu_z}}{\|\mathscr{G}_\mu\|}, \rho = \sigma \|\mathscr{G}_\mu\| \sqrt{w} \tag{13}$$

Objects are often approximately polyhedral, with each facet having at least one close Gaussian for which only this facet is close. This implies that such models may be reconstructed from the FV even for very large T.

c. Point cloud reconstruction from FV representation using a deep decoder - A direct expression for reconstructing a point cloud from its FV representation is not available for $K > 1$. For illustration, we show now such a reconstruction obtained with a deep decoder, taking FV as an input and providing a point cloud. We consider a special FV, associated with a GMM with Gaussian centered on a grid; see Section III-D below. The decoder architecture is identical to the convolutional part of the classification network presented in Section III-E followed by two fully connected layers: FC(T),FC($3T$). The loss function between the original and the reconstructed point sets, S_1 and S_2, should be invariant to point order. We use a loss function that is the sum of Chamfer distance and the Earth mover's distance, which were used (separately) in [41]. Figure 2 shows a qualitative comparison between the original point cloud and the reconstructed point cloud. It shows that the decoder captured the overall shape while not positioning the points exactly in their original position.

Fig. 2. Point cloud reconstruction from FV representation using a deep decoder. The original (left), and the reconstructed point cloud (right).

C. Generalizing Fisher vectors to 3D modified Fisher vectors

We propose to generalize the FV along two directions:

Choice of the mixture model - Originally, the mixture model was defined as a maximum likelihood model. This makes the model optimally adapted to the training data and gives the Fisher representation of each Gaussian the nice property of being sensitive only to the deviation from the average training data. It is not the only valid option, however, and not a good choice if we prefer to maintain a grid structure. Therefore, in the proposed generalization we shall use other mixture models that rely on Gaussian grids.

Choice of the symmetric function - As apparent from eq. (3), the components of the Fisher vector are sums over all input points, regardless of their order and any structure they create. They are *symmetric* in the sense proposed in [30] and are therefore adequate for representing the orderless and structureless set of points. Note also that any other symmetric function, applied over the summands in eq. (3), would also induce a vector that can represent orderless sets. We will propose such functions and use them instead of or in addition to the Fisher vector sums.

D. The proposed 3DmFV generalization

Changing the mixture model - For the underlying density model, we use a mixture of Gaussians with Gaussian centers (μ_k) on a uniform 3D $m \times m \times m$ grid. Such Gaussians induce a Fisher vector that preserves the point set structure: the presence of points in a specific 3D location would significantly influence only some, pre-known, Fisher components. The other GMM parameters, weight and covariance, are common to all Gaussians. The weights are selected as $w_k = \frac{1}{K}$ and the covariance matrix as $\Sigma_k = \sigma_k I$ with $\sigma_k = \frac{1}{m}$. All points are contained in the unit sphere. The uniformity is essential for shared weight (convolutional) filtering. The size of the mixture model is moderate and ranges from $m = 3$ to 9.

The proposed uniform model is not as effective as the maximum likelihood model for representing the distribution of point clouds. Recall, however, that the GMM does not represent a

specific model or a specific class but rather the average model, which is much closer to uniform. The inaccuracy is more than compensated for by the power of the convolutional network, as we shall see in the comparison between the different models.

Changing/Adding other symmetric functions - For Fisher vectors, the sum is used as a symmetric function. While the sum is asymptotically optimal, it does not give full information about the input points for finite point sets. For small point sets the Fisher vectors may be invertible, as suggested above, implying that the FVs implicitly carry the full information about the set. For the practical case of large finite point sets, we propose to add information. To maintain the order independence, other features should be symmetric as well.

We experimented with several options for additional symmetric functions, and eventually chose the maximum and minimum functions. Note that the maximum was also used in [30] as a single summarizing feature for each of the learned features. Thus, the components of the proposed generalized vector, denoted 3DmFV, are given in eq. 14 and obtained as follows: each component is either a sum, max, or min function, evaluated on the set of one gradient component. In our experiments we found that partial, more compact, representations (especially those focusing on the minimum and maximum function) may lead to improved accuracy, and we describe them as well. However, we also found that the minimal weight derivative is always a constant and omitted this specific function. The minimal value associated with a specific Gaussian corresponds to the farthest point and its $\gamma_t(k)$ value, which is 0 in practice.

$$\mathscr{G}^X_{3DmFV_\lambda} = \begin{bmatrix} \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(\boldsymbol{p}_t)\Big|_{\lambda=\alpha,\mu,\sigma} \\ \max_t (L_\lambda \nabla_\lambda \log u_\lambda(\boldsymbol{p}_t)|_{\lambda=\alpha,\mu,\sigma} \\ \min_t (L_\lambda \nabla_\lambda \log u_\lambda(\boldsymbol{p}_t))|_{\lambda=\mu,\sigma} \end{bmatrix} \quad (14)$$

For $K = m^3$ Gaussians, there are therefore $20 \times K$ components in the 3DmFV representation ($20 = 3(3+3)+2$). The 3DmFV is best visualized as a $20 \times K$ matrix. Figure 3 depicts a point cloud (right) and its 3DmFV representation ($m = 5$) as a color coded image (left). Each column of the image represents a single Gaussian in a $5\times5\times5$ Gaussian grid. Zero values are white whereas positive and negative values correspond respectively to the red and blue gradients. Note that the representation lends itself to intuitive interpretation. For example, many columns are white, except for the first two top entries. These correspond to Gaussians that do not have model points near them; see eq. 6.

Normalization - Following [36] (Sec. 2.3), we applied two normalizations on the 3DmFV representation: First, we applied an element-wise signed square root normalization, and then an L2 normalization over all 20 vectors corresponding to all Gaussians and a single feature λ_i. This last normalization equalizes the derivatives with respect to different parameters.

Time complexity - Given the above operations which define the 3DmFV representation, its theoretical time complexity is linear w.r.t the number of Gaussians and the number of points, i.e. it is $O(KT)$.

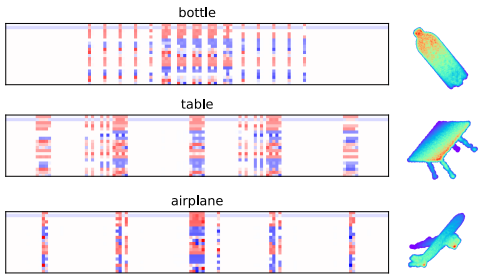

Fig. 3. 3DmFV representation (left) and the corresponding point cloud (right)

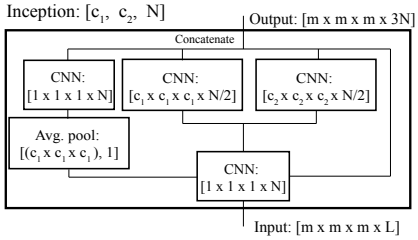

Fig. 4. Inception module used in 3DmFV-Net

E. 3DmFV-Net classification architecture

The proposed network receives a point cloud and converts it to a 3DmFV representation on a grid. The main parts of the network consist of an Inception module [42], visualized in Figure 4, maxpooling layers, and finally four fully connected layers. The network output consists of classification scores; see Figure 1. The network is trained using back propagation and standard softmax cross-entropy loss with batch normalization after every layer and dropout after each fully connected layer. The network has approximately 4.6M trained parameters, the majority of which are between the last maxpooling layer and the first fully connected layer.

IV. EXPERIMENTS

We first evaluate the classification performance of our proposed 3DmFV-Net and compare it to previous approaches. We then evaluate several variants of the proposed representation. Finally, we analyze our method's robustness to noise.

A. Implementation

Datasets - We evaluate 3DmFV-Net classification algorithm on data from two different domains:

1) *CAD data - ModelNet*: The ModelNet40 dataset [4] consists of 12311 CAD models from 40 object categories split into 9843 for training and 2468 for testing. The data, represented as triangle mesh, is sampled as described in [30] in order to generate point clouds. We also experimented with the ModelNet10 dataset, which contains 4899 CAD models from 10 object classes split into 3991 for training and 908 for testing.
2) *LiDAR data - Sydney Urban Objects*: The Sydney Urban Objects dataset[1] contains Velodyne LiDAR scans

[1]http://www.acfr.usyd.edu.au/papers/SydneyUrbanObjectsDataset.shtml

TABLE I
CLASSIFICATION ACCURACY ON MODELNET40 AND MODELNET10

Method	ModelNet10	Modelnet40
MVCNN [43]	-	90.1
3DShapeNets [4]	83.5	77.32
VoxNet [3]	92.0	83.0
OctNet [21]	90.9	86.5
O-CNN [22]	-	89.9
VRN [20]	93.6	91.33
VRN ensemble [20]	97.14	95.54
FusionNet [44]	93.1	90.8
PointNet [30]	-	89.2[a]
PointNet++ [31]	-	90.7[a]
Kd-network [32]	94.0[b] / 93.3[a]	91.8[b] / 90.6[a]
3DmFV+VoxNet	94.3	88.5[c]
Our 3DmFV-Net	95.2[ac]	91.6[c]/91.4[a]

*The point based methods use [a]1024,[b]32768, [c]2048 points.

TABLE II
CLASSIFICATION PERFORMANCE ON SYDNEY DATASET

Method	F1 score
Triangle+SVM [16]	0.67
GBH+SVM [15]	0.71
VoxNet [3]	0.72
PointNet [30]	0.7
3DmFV-Net (no aug.)	0.73
3DmFV-Net (rotation aug.)	**0.76**

TABLE III
F1 SCORES PER-CLASS

class	4wd	bldg.	bus	car	ped.	pillar	pole	tfc. light	tfc. sign	tree	truck	trunk	ute	van
num. ins.	21	20	16	88	152	20	21	47	51	34	12	55	16	35
F1	0.22	0.64	0.21	0.81	0.99	0.84	0.68	0.74	0.78	0.82	0.27	0.74	0.31	0.57

of 14 object classes with a total of 588 objects over four train/test splits. The classes are highly imbalanced, therefore we report the average F1 score, weighted by class support.

Training details: Unless otherwise specified, the 3DmFV-Net (Figure 1) was trained using an Adam optimizer with a learning rate of 0.001 with a decay of 0.7 every 20 epochs. The point cloud (of 2K points) is centered around the origin and scaled to fit a cube of edge length 2. Data is augmented using random anisotropic scaling (range: $[0.66, 1.5]$) and random translation (range: $[-0.2, 0.2]$) in each axis, similarly to [32]. We used Tensorflow on a NVIDIA Titan Xp GPU. Training took $\sim$ 7h.

B. Classification performance

Table I compares 3DmFV-Net with previous approaches on the ModelNet40 and Modelnet10 datasets. The comparison metric is instance accuracy. Clearly, the proposed method wins over most methods. It is comparable to the Kd-network, which requires a much larger input ($\sim 32K$ points) and is not very robust to rotations and noise [32]. It is less accurate only than the more complex VRN ensemble method [20], which operates on voxelized input and averages 6 models, each trained for 6 days. However, it is slightly better than a single VRN model. Note also that direct comparison is somewhat unfair because, unlike the voxelized description, point based methods (like ours) do not have direct access to the mesh.

We also tested a combination of the 3DmFV representation ($m = 8$) with the simpler convolutional network that mimics the one used in VoxNet [3]. Although this combination (denoted 3DmFV+VoxNet) uses a lower resolution grid than VoxNet (32^3 voxels), it is more accurate. Thus, the performance boost of the 3DmFV-Net may be attributed to both the representation and the architecture.

Table II compares the average F1 score of 3DmFV-Net to Voxnet [3] and other descriptor based SVM classifiers [15], [16] on the Sydney dataset. We also tested PointNet on it. This required some adaptation to enable PointNets publicly available code to accept point clouds with varying number of points. It shows that without any data augmentations or voting strategies during training, as suggested by VoxNet, we achieve competitive results. When rotating the input models at train time by $360/n$ degrees, where $n = 4$ the performance increases significantly. Table III shows the F1 score per class alongside the number of instances in each class. It shows that the score improves when more instances are available for training (e.g. pedestrian and car) but may also be high when the class is of unique shape.

C. Testing variations of the 3DmFV representation

We now test partial, more compact variants of the proposed representations. The first variant is the Fisher vectors. The 3DmFV generalization of FV uses different symmetric functions (and not only the sum, as in FV), and different GMMs. We considered the combinations of several symmetric functions, with GMMs obtained either from the maximum likelihood (ML) optimization obtained by the expectation maximization (EM) algorithm or from Gaussians on a 3D $5\times5\times5$ grid. The tested symmetric functions include maximum (3DmFV-max), minimum (3DmFV-min), and sum of squares (3DmFV-ss). We tested these combinations with a nonlinear classifier (4 fully connected layers of sizes $(1024, 256, 128, 40)$ with ReLU activation). For reference to the original FVs, we tested the ML GMM with a linear classifier as well. All models were represented by 1024 points. Table IV reveals that the 3DmFV representation always wins over the FV representation, and that using grid GMM is comparable to the optimal ML GMM. Using maximum or minimum as a symmetric function yields comparable results with fewer parameters. Note that with the convolutional network, possible only with grid GMM, the accuracy is much higher (Table I).

Resolution and standard deviation We found that accuracy increases with both grid size (m) and the number of points

TABLE IV
CLASSIFICATION ACCURACY OF 3DMFV VARIATIONS

Rep.	ML + LinCls	ML + NonLinCls	Grid + NonLinCls
FV	58.4	82.8	84.5
3DmFV-ss	58.8	85.0	84.4
3DmFV-min	67.7	87.7	86.1
3DmFV-max	68.6	87.4	85.3
3DmFV	76.8	88.0	87.7

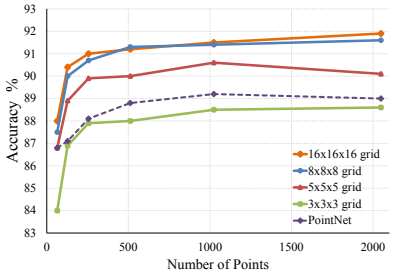

Fig. 5. Effects of grid resolution and number of input points on the evaluation accuracy.

representing the model; see Figure 5. Note that performance saturates in both parameters. PointNet [30] seems to be more sensitive to the number of points. This is probably because their descriptors are learned as well. The architecture is slightly different for each grid size. We also found that the performance is insensitive to the selection of standard deviation as long as it is not too small. In the case of small σ, most points do not contribute to any Gaussian, rendering the FV representation empty.

D. Time performance

All time measurements were averaged over 2448 point clouds divided into batches of 16 samples on a Titan Xp GPU. We start with the computation time of the 3DmFV representation, and show empirically in Figure 6 (left), a linear computation time increase w.r.t the number of points for each different grid size $K = 3^3, 5^3, 8^3$. As expected, we also found that the run time is linear in the number of Gaussians. Next we measure the total inference time and compare with other methods. The results are shown in Figure 6 (right). Note that the total inference time includes the representation computation time which is the only contributor to the time increase since the network classification inference time is approximately constant. Also note that VoxNet time was measured on a 32^3 voxel grid without factoring the voxelization. It is reported as a constant reference since it does not operate directly on point clouds. The results show a trade-off between accuracy and speed. Despite the computation overhead of the 3DmFV representation, it operates in real-time and scales linearly while allowing the flexibility of adjusting the grid size based on accuracy/speed requirements.

These results makes 3DmFV-Net an appealing choice for both sparse and dense point clouds since it provides high accuracy for low density point clouds, and fast run-time for higher density point clouds. For faster speeds, a high density cloud may be down sampled with a small impact on accuracy.

E. Robustness evaluation

To simulate real-world point cloud classification challenges, we tested 3DmFV-Net's robustness on the Modelnet40 dataset under several types of noise:

Uniform point deletion - randomly deleting points is equivalent to classifying clouds that are smaller than those used for training.

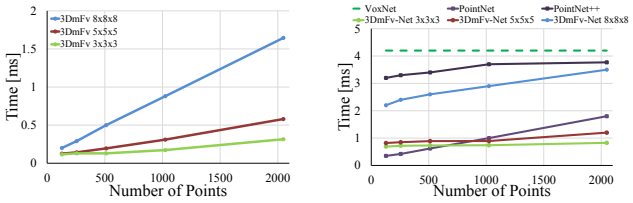

Fig. 6. Run-time of 3DmFV-Net compared to state-of-the-art. 3DmFV representation computation time (left), and the total classification inference time (right).

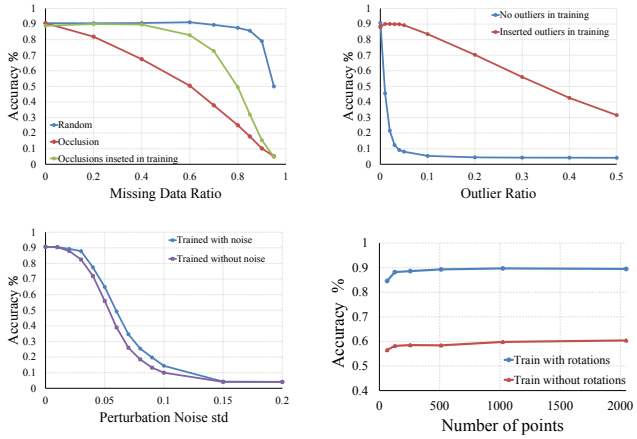

Fig. 7. 3DmFV-Net robustness to data corruptions. Classification accuracy results for missing data (top-left), outlier insertion (top-right), perturbation noise (botom-left), and rotations (bottom-right).

Focused region point deletion - selecting a random point and deleting its closest points (the number of points is defined by a given ratio), simulating occlusions.

Outlier points - adding uniformly distributed points.

Perturbation noise - adding small translations, with a bounded Gaussian magnitude, independently to all points, simulating measurement inaccuracy.

Random rotation - randomly rotating the point cloud w.r.t. the global reference frame, simulating the unknown orientation of a scanned object.

The results (Figure 7) demonstrate that the proposed approach is inherently robust to perturbation noise and uniform point deletions. For the other types of data corruptions, training the classifiers with any of these types of noise made it robust to them.

V. Conclusion

In this work, we propose a new 3D point cloud representation, the 3D modified Fisher vector. It preserves the raw point cloud data while using a grid for structure. This allows the use of the proposed CNN architecture (3DmFV-Net) for efficient and accurate object classification for 3D sensor data in real-time applications such as autonomous driving or 6DOF pose estimation for robotic grasping.

Representing data by non-learned features goes against the deep network principle that the best performance is obtained only by learning all components of the classifier using an end-to-end optimization. The proposed representation achieves state of the art results relative to all methods that use point

cloud input, and therefore provides evidence that end-to-end learning is not always essential.

REFERENCES

[1] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[2] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *Robotics Research*. Springer, 2017, pp. 235–252.

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

[4] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[5] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.

[6] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007, pp. 1–8.

[7] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision*, vol. 105, no. 3, pp. 222–245, 2013.

[8] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *The IEEE International Conference on Robotics and Automation(ICRA)*. IEEE, 2009, pp. 3212–3217.

[9] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[10] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European Conference on Computer Vision*. Springer, 2010, pp. 356–369.

[11] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3d local surface description and object recognition," *International Journal of Computer Vision*, vol. 105, no. 1, pp. 63–86, 2013.

[12] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, "Our-cvfh–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation," in *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*. Springer, 2012, pp. 113–122.

[13] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3d object classification," in *The IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2011, pp. 2987–2992.

[14] Z.-C. Marton, D. Pangercic, N. Blodow, and M. Beetz, "Combined 2d–3d categorization and classification for multimodal perception systems," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1378–1402, 2011.

[15] T. Chen, B. Dai, D. Liu, and J. Song, "Performance of global descriptors for velodyne-based urban object recognition," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 667–673.

[16] M. De Deuge, A. Quadros, C. Hung, and B. Douillard, "Unsupervised feature learning for classification of outdoor 3d scans," in *Australasian Conference on Robitics and Automation*, vol. 2, 2013, p. 1.

[17] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3d local feature descriptors," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 66–89, 2016.

[18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, 2017, pp. 4278–4284.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[20] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," *arXiv preprint arXiv:1608.04236*, 2016.

[21] G. Riegler, A. O. Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 3, 2017.

[22] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 72, 2017.

[23] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1355–1361.

[24] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv preprint arXiv:1706.01307*, 2017.

[25] T. Hackel, M. Usvyatsov, S. Galliani, J. D. Wegner, and K. Schindler, "Inference, learning and attention mechanisms that exploit and preserve sparsity in convolutional networks," *arXiv preprint arXiv:1801.10585*, 2018.

[26] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[27] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 37–45.

[28] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks," in *Computer Graphics Forum*, vol. 34. Wiley Online Library, 2015, pp. 13–23.

[29] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 3189–3197.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[31] 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*, 2017, pp. 5105–5114.

[32] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[33] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2003, pp. 2743–2748.

[34] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility," *Journal of field robotics*, vol. 23, no. 10, pp. 839–861, 2006.

[35] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[36] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," *Computer Vision–ECCV 2010*, pp. 143–156, 2010.

[37] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing Systems*, 1999, pp. 487–493.

[38] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep fisher networks for large-scale image classification," in *Advances in Neural Information Processing Systems*, 2013, pp. 163–171.

[39] F. Perronnin and D. Larlus, "Fisher vectors meet neural networks: A hybrid classification architecture," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3743–3752.

[40] J. Krapac, J. Verbeek, and F. Jurie, "Modeling spatial layout with fisher vectors for image categorization," in *The IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 1487–1494.

[41] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3d object reconstruction from a single image," *arXiv preprint arXiv:1612.00603*, 2016.

[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[43] 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 (CVPR)*, 2015, pp. 945–953.

[44] V. Hegde and R. Zadeh, "Fusionnet: 3d object classification using multiple data representations," *arXiv preprint arXiv:1607.05695*, 2016.