Deep Learning Advances in Computer Vision with 3D Data: A Survey

ANASTASIA IOANNIDOU, ELISAVET CHATZILARI, SPIROS NIKOLOPOULOS,
and IOANNIS KOMPATSIARIS, Information Technologies Institute (ITI),
Centre for Research and Technology Hellas (CERTH)

Deep learning has recently gained popularity achieving state-of-the-art performance in tasks involving text, sound, or image processing. Due to its outstanding performance, there have been efforts to apply it in more challenging scenarios, for example, 3D data processing. This article surveys methods applying deep learning on 3D data and provides a classification based on how they exploit them. From the results of the examined works, we conclude that systems employing 2D views of 3D data typically surpass voxel-based (3D) deep models, which however, can perform better with more layers and severe data augmentation. Therefore, larger-scale datasets and increased resolutions are required.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computing methodologies** → **Computer vision tasks**; **Scene understanding**; **3D imaging**; **Neural networks;**

Additional Key Words and Phrases: 3D data, 3D object recognition, 3D object retrieval, 3D segmentation, convolutional neural networks, deep learning

ACM Reference Format:
Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. 2017. Deep learning advances in computer vision with 3D data: A survey. ACM Comput. Surv. 50, 2, Article 20 (April 2017), 38 pages.
DOI: http://dx.doi.org/10.1145/3042064

1. INTRODUCTION

Recent advancements in 3D sensing technology (e.g., LIDAR and UAV sensors) and the appearance of low-cost devices such as Microsoft Kinect have made the collection of 3D data more feasible and affordable than ever. Based on the scanning device employed for capturing the 3D scene or object of interest, raw data are collected in different forms. UAV scanners get range images from different camera viewpoints. Then, these images are typically combined through a registration process or Structure-from-Motion (SfM) techniques in order to discard noisy data, establish correspondences between them, and ultimately generate a unified 3D point cloud for further processing. LIDAR scanners provide a 3D point cloud model of the captured scene. Additionally, digital images can also be acquired during capturing in order to enhance the quality of the final point cloud. Kinect-like devices, from the other side, capture RGB-D images, that is, one color (RGB) and one depth image from each camera viewpoint that can either be

This work is supported by the EU Horizon 2020 Programme (H2020/2015-2018) under grant agreement 665066 corresponding to project DigiArt (The Internet Of Historical Things And Building New 3D Cultural Worlds, http://digiart-project.eu/).
Authors' addresses: A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, Multimedia Knowledge and Social Media Analytics Laboratory (MKLab), Information Technologies Institute (ITI), Centre for Research and Technology Hellas (CERTH), 6th km Charilaou - Thermis Rd, P.O. Box 60361, 57001, Thessaloniki, Greece; emails: {ioananas, ehatzi, nikolopo, ikom}@iti.gr.

DOI: http://dx.doi.org/10.1145/3042064

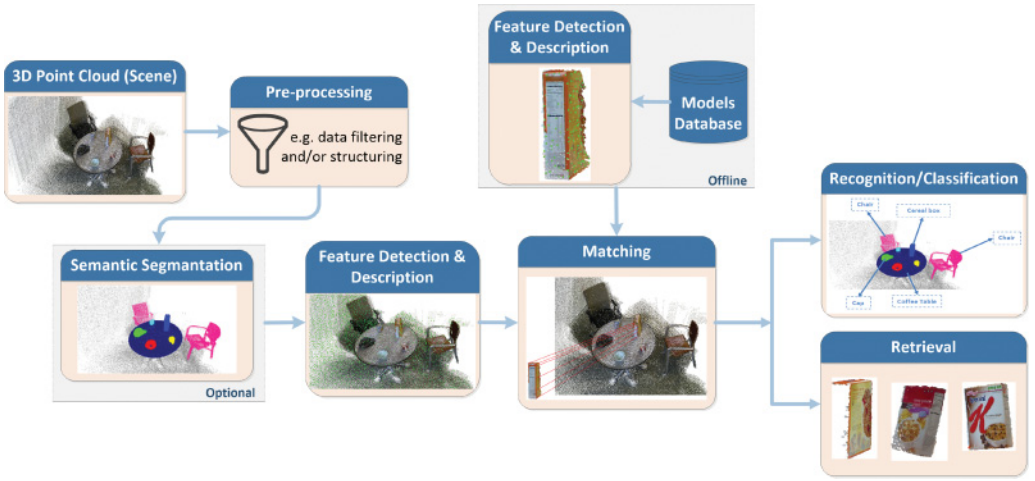


Fig. 1. Diagram of the core analysis steps typically applied for 3D data manipulation.

used for creating a point cloud representation or as separate input channels. Finally, hyperspectral cameras provide images from a specific range of spectral bands, forming 3D hypercubes.

The increasing abundance of 3D data encouraged the research community to exploit this richer content for addressing several computer vision problems related to understanding 3D scenes, for example, 3D Object Classification, 3D Object Recognition, and 3D Shape Retrieval. Indeed, the possibility of using the additionally provided attributes of depth and full 3D geometry represents an important advantage that can significantly boost the performance of several applications. A generic pipeline for processing a 3D scene is depicted in Figure 1. Scans of real scenes can contain millions of points, therefore at first, some form of preprocessing is commonly applied. Point reduction (i.e., remove redundant points in order to reduce the computational cost), data structuring (i.e., organize the point cloud using data structures like kd-trees or octrees), and hardware exploitation (e.g., GPU calculations) are a few of the methods proposed so far. Following, segmentation of the point cloud is typically performed in order to identify semantically meaningful regions. After acquiring the objects of the scene from segmentation, keypoint detection and descriptors extraction are applied to every identified object or scene segment. The extracted representation is subsequently utilized in order to match the scene segments with known object models and finally recognize or classify them into a category or even retrieve similar objects.

Until recently, the standard approaches to many vision tasks involved the extraction of handcrafted local or global descriptors from the available images or videos. The breakthrough results reported by Krizhevsky et al. [2012] in the 2012 ILSVRC[1] image classification task though, completely changed the landscape of computer vision. Neural networks and Deep Learning (DL) now dominate on almost every recognition and detection task making every group of computer vision researchers and practitioners redesign their systems. Although neural networks have a long history [McCulloch and Pitts 1943; Rosenblatt 1958], the rapid evolution of powerful GPUs and the availability of really large datasets are considered the main reasons for their recent success.

After DL established its dominance in 2D computer vision problems, it was only a matter of time before it was adapted for addressing tasks in more complicated settings,

[1]ImageNet LSVRC: http://www.image-net.org/challenges/LSVRC/.

such as dealing with 3D input data. Driven by this evolution, this article covers the main literature and provides an overview of existing methods proposed for tackling 3D computer vision tasks via DL. In order to make the article more self-contained, related work on 3D data manipulation is provided first, followed by a short introduction to deep networks and the architectures used so far in 3D computer vision tasks. Afterward, the main section of the article regarding the employment of DL in the 3D domain is presented. Finally, conclusions are discussed.

2. RELATED WORK

In this section, a brief review of the main techniques employed so far in 3D computer vision is provided following the workflow depicted in Figure 1. More specifically, we start from works related to 3D scene segmentation and proceed with 3D keypoint detection and 3D descriptor extraction methods. Then, 3D shape retrieval and 3D object recognition methodologies are presented. The cited works have also been included in a tabular form for easier reference and can be found in the electronic appendix. Additionally, the exploitation of time in relation to 3D data is examined in a separate section discussing 4D modeling. Finally, reviews about DL and its application to the 2D domain found in the literature are reported.

2.1. 3D Scene Segmentation

3D scene segmentation includes labeling each point of a scene as part of a foreground object of interest or of a background surface. This task is also known as semantic segmentation/labeling and it is closely related to 3D object classification, that is, correctly identifying the class in which a 3D object belongs. Inspired from the techniques used in the 2D/image domain, several methods ranging from region growing (e.g., Vo et al. [2015]) and graph-based approaches (e.g., Sima and Nuchter [2013]) to superpixels (e.g., Papon et al. [2013] and Aijazi et al. [2013]) have been considered for segmenting 3D data. Furthermore, clustering algorithms [Douillard et al. 2011], 3D Hough Transform [Borrmann et al. 2011], RANSAC [Schnabel et al. 2007], and probabilistic models, such as Markov Random Fields (MRFs) and Conditional Random Fields (CRFs), have also been employed (e.g., Valentin et al. [2013]). Some of the proposed methods process RGB-D data, that is, separate RGB and depth images also referred to as 2.5D data, in order to segment the scene but there is also significant, recent work applied directly to 3D point clouds. In Zelener [2015], eight systems for 3D object classification and semantic segmentation were compared in terms of classification accuracy, utilization of 3D data, performance on specific object categories and the techniques employed for segmentation, descriptors extraction, and ultimately classification. Finally, a survey of 3D point cloud segmentation methods is presented in Nguyen and Le [2013]. The authors divided 3D point cloud segmentation methods into five categories, namely, *region -based*, *edge -based*, *attributes -based*, *model -based*, and *graph -based* methods.

2.2. 3D Keypoint Detection and 3D Descriptors Extraction

3D keypoint detection is a critical step of an object recognition or retrieval pipeline. Inspired from 2D feature engineering, several 3D keypoint detectors have been proposed. Tombari et al. [2013] categorized the existing approaches into two classes: (1) *fixed-scale keypoint detectors*, which identify distinctive keypoints at a constant scale given to the algorithm as an input argument, and (2) *adaptive-scale keypoint detectors*, which identify keypoints after creating a scale space defined on the surface or alternatively after computing an embedding of the data on a 2D plane. Indicative examples of *fixed-scale* detectors are Local Surface Patches (LSPs) [Chen and Bhanu 2007], Intrinsic Shape Signatures (ISSs) [Zhong 2009], and the 3D detector proposed in Mian et al. [2010] termed as "KeyPoint Quality" (KPQ) in Tombari et al. [2013],

while a popular *adaptive-scale* detector is MeshDoG [Zaharescu et al. 2009]. Following the widely accepted methodology applied for comparing 2D detectors [Schmid et al. 2000; Mikolajczyk et al. 2005], several studies evaluating 3D detectors have recently been conducted, for example, Tombari et al. [2013] and Filipe and Alexandre [2014]. In Tombari et al. [2013], KPQ was found as an appropriate detector for 3D object recognition and 3D shape retrieval based on a thorough comparison of different detectors by matching descriptors computed at the extracted keypoints. Filipe and Alexandre [2014] experimented on a RGB-D dataset in order to investigate the detectors robustness to rotations, scale changes, and translations.

Extensive work has also been done in the area of 3D descriptors. A distinction between local and global approaches is also used in 3D data just like in the 2D domain. Some widely employed local descriptors are Spin Image [Johnson and Hebert 1999], 3D Shape Context (3DSC) [Frome et al. 2004], Point Feature Histogram (PFH) [Rusu et al. 2008], Signature of Histogram of Orientations (SHOT) [Tombari et al. 2010], and Rotational Projection Statistics (RoPS) [Guo et al. 2013]. Popular global descriptors are Ensemble of Shape Functions (ESF) [Wohlkinger and Vincze 2011] and Viewpoint Feature Histogram (VFH) [Rusu et al. 2010]. A comparative evaluation of 3D local and global descriptors in the restricted context of object and category recognition was performed by Alexandre [2012]. This study highlighted the importance of color for object and category recognition since the top two descriptors were color variants of the PFH and SHOT descriptors. This observation is also consistent with the findings of Filipe et al. [2015]. Moreover, a thorough investigation on the performance of several 3D local descriptors in the context of 3D object recognition, 3D modeling, and 3D shape retrieval is available in Guo et al. [2016a]. The authors based their study on the popular methodology proposed by Mikolajczyk and Schmid [2005] for the evaluation of 2D local descriptors. SHOT was identified as a good choice for applications dealing with big point clouds while time is a critical factor. The recently proposed RoPS descriptor [Guo et al. 2013] achieved the most stable performance across different datasets. A study in order to identify the best 3D detector/descriptor pair for shape registration, object recognition, and shape retrieval was performed by Salti et al. [2012]. The combination of ISS detector with the SHOT descriptor was one of the best on both retrieval and recognition scenarios. Recently, TriSI, a new 3D descriptor inspired by the spin image descriptor, was proposed [Guo et al. 2015]. TriSI was proven to perform better than other state-of-the-art descriptors like SHOT [Tombari et al. 2010] and RoPS [Guo et al. 2013] in the task of 3D object recognition.

2.3. 3D Object Retrieval and Recognition

Content-based image retrieval is a well-studied task in computer vision. In the 3D domain, shape retrieval is one of the first problems that attracted the attention of the research community. Given a 3D object query, the goal is to retrieve semantically similar objects from a given database. Two steps are included in a typical retrieval pipeline: (a) descriptors extraction from the 3D objects, and (b) matching of the queries' descriptors with the stored descriptors of the database objects using an appropriate similarity measure. Existing approaches on 3D object retrieval can be divided into [Gao and Dai 2014] (a) *3D model-based methods*, which are based mostly on low-level descriptors extraction from the 3D models, and (b) *view-based methods*, which utilize multiple 2D views of the 3D objects. Certain geometrical and topological properties of the 3D objects have been exploited for low-level descriptors extraction, while the Bag-of-Words (BoW) framework, widely utilized in the 2D domain, is also used in 3D (e.g., Lavoue [2012]). Popular view-based 3D object retrieval methods include the Light Field Descriptor (LFD) [Chen et al. 2003] and the Compact Multi-View Descriptor (CMVD) [Daras and Axenopoulos 2010]. In the work of Gao et al. [2011], a query view

selection approach for interactive 3D object retrieval is proposed leading to increased performance. Examples of pioneer content-based search and retrieval methodologies for 3D objects include Kazhdan et al. [2003], Zarpalas et al. [2006], Mademlis et al. [2009], and Makantasis et al. [2016]. Recently, Gao et al. [2012] presented an effective method for comparing multiple views of 3D objects based on hypergraph analysis. Several surveys on 3D object retrieval are available [Bustos et al. 2007; Tangelder and Veltkamp 2007; Liu 2012]. Moreover, a benchmarking contest, named SHREC,[2] is conducted every year since 2006 to examine the performance of 3D object retrieval algorithms. During the last few years, SHREC organizes multiple tracks under different retrieval scenarios, for example, sketch-based 3D object retrieval. A comparison of methods addressing this task can be found in Li et al. [2014a].

The task of correctly identifying specific objects that appear in a 3D scene, usually in a depth/range image, and estimating the location and orientation of each object (i.e., recover their poses) is known as 3D Object Recognition. Traditional recognition approaches generally comprise two phases: (a) keypoints detection and descriptors extraction, and (b) surface matching. Based on the type of descriptors used in the pipeline, recognition methods are often characterized as global or local [Guo et al. 2014; Aldoma et al. 2012b]. Local descriptors-based methods (e.g., Aldoma et al. [2012a], Tombari and Di Stefano [2012], and Guo et al. [2013]) exploit local detectors and descriptors in order to extract a rich representation at first from all available object models and afterward, from the examined scene(s). Ultimately, object hypotheses are generated by establishing correspondences between the scene and every object model through descriptor matching. Verification of the generated hypotheses is further conducted in order to accurately specify the objects appearing in the scene. A recent survey on 3D object recognition based on local descriptors is available in Guo et al. [2014]. On the other hand, global descriptors-based methods (e.g., Rusu et al. [2010], Aldoma et al. [2012b], Tang et al. [2012], and Wang et al. [2013]) employ global descriptors and require segmentation of the scene in candidate objects, since only one descriptor is computed for each input point cloud. Object hypotheses generation and verification are performed in these methods as well. After identifying the objects present in each scene, the pose of each object needs to be determined. The most popular technique used for pose refinement is the Iterative Close Point (ICP) algorithm [Besl and McKay 1992].

2.4. 4D Modeling

Time is an important factor in many applications, but all the aforementioned 3D tasks do not take it into account. 4D modeling focuses on capturing 3D data and their progress over time. Interesting works in this domain are related to construction planning and cultural heritage objects/sites. In the work of Khatib et al. [2007], it was highlighted that 4D modeling could benefit the planning process in the construction industry in terms of cost, quality, and time. Bosche et al. [2010] exploited laser scanning and 4D modeling and designed a system for automated construction progress control. The system could recognize elements that were present in a 3D scan and estimate the progress based on the construction plan. A framework for 4D reconstruction of cultural heritage content from multiple web-retrieved images was presented in Ioannides et al. [2013], Doulamis et al. [2013], and Kyriakaki et al. [2014]. Time-varying 3D models were constructed using dense image matching methods, motion information, and tracking. The result of the retrieval and reconstruction process was demonstrated via a visualization tool allowing the user to interact with it. Charbonneau et al. [2015] proposed a method for 4D modeling in order to offer to a museum's visitors the opportunity to view a cultural heritage site at different points in time and also

[2]SHREC Contest: http://www.shrec.net/.

access related information through an interactive interface created with the Unity3D[3] game engine. Another interesting application can be found in Schindler and Dellaert [2012] where 4D city models were constructed from historical urban photographs and information using a SfM framework. A 4D city viewer was also presented allowing the user to interact with the 3D models in several ways. Finally, Doulamis et al. [2015] proposed a 4D land information management system where dynamic change historic maps were built allowing the identification of surfaces requiring more careful reconstruction as the next time instances became available to the system. Visualization was also supported for manipulating the 4D models and their semantics.

2.5. Current Trends

During the last few years, a gradual transition from traditional computer vision methods to DL-based techniques has been observed. Starting from 2012, more and more works addressing computer vision tasks involving text (e.g., Kalchbrenner et al. [2014]), speech (e.g., Graves et al. [2013]), images (e.g., Ren et al. [2015]), and even graphs (e.g., Niepert et al. [2016]) via DL are being published every year. Recently, a few review papers have also become available [Deng 2014; LeCun et al. 2015; Guo et al. 2016b; Schmidhuber 2015]. In Deng [2014], several DL techniques were presented and a categorization of them into three classes, namely, discriminative, generative, and hybrid, was provided. Moreover, historical facts were reported and several applications of DL were reviewed. LeCun et al. [2015] elaborated on the key aspects of DL and focused on Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and relevant applications. In Guo et al. [2016b], four deep architectures (CNNs, Restricted Boltzmann Machines (RBMs), Autoencoders (AEs) and Sparse Coding) were analyzed and developments made in many tasks were summarized. In addition, challenges and recent trends were discussed. Finally, an in-depth survey about neural networks and DL was conducted and presented in a timeline format by Schmidhuber [2015].

Despite the rich content provided by the review papers on DL cited in the previous paragraph, all of them report developments involving solely 1D and/or 2D data (i.e., text, sound, images) and do not examine at all the 3D case. This article contributes to this void in the literature by reviewing the set of solutions that are based on architectures of deep neural networks (DNNs) and have been proposed to address 3D computer vision tasks.

3. BACKGROUND ON DNNS

DL is a subfield of machine learning, recently attracting a great deal of attention mainly due to the remarkable winning performance of the AlexNet model [Krizhevsky et al. 2012] in the 2012 ImageNet challenge. Ever since, several architectures of DNNs have been adopted and state-of-the-art results are constantly being reported in tasks such as image classification, speech recognition, natural language processing, and many others. A standard neural network model consists of distinct layers of units (i.e., neurons) that are connected in an acyclic graph. Units get activated through weighted connections between them. Typically, the model contains an input and an output layer, while any number of hidden layers can be added in-between. Modern deep architectures include models with more than 10 hidden layers, while both feedforward (acyclic) and recurrent (cyclic) neural networks are widely adapted.

DL is all about learning hierarchies of data representations. Starting from raw data, every layer of a network converts the data into a more abstract representation. In order

[3]Unity3D: https://unity3d.com/.

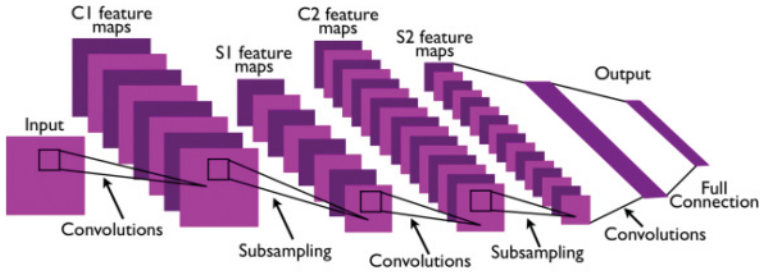
to exhibit the desired behavior, appropriate weights have to be assigned. The most popular algorithm for learning the weights of a neural network is a gradient descent method named backpropagation [Rumelhart et al. 1986], originally developed during the 1960s and 1970s. Given the computational power of that time, training a deep network with backpropagation was in practice a slow, difficult process. In the late 1990s, LeCun et al. [1998] managed to successfully apply Stochastic Gradient Descent (SGD) via backpropagation in order to train CNNs (i.e., an example of feedforward networks) for handwritten digit recognition. However, the lack of significant computational power from one side and of big amounts of labeled data from the other hindered the widespread use of CNNs. In the meantime, alternative methods with robust theoretical foundations such as Support Vector Machines (SVMs) [Cortes and Vapnik 1995] started to achieve remarkable performance and the interest in deep networks inevitably decreased. On the 2012 ILSVRC though, deep neural networks' power was eventually acknowledged. Krizhevsky et. al. did not just win the competition but more importantly changed completely the computer vision's community perspective.

3.1. DNNs Architectures

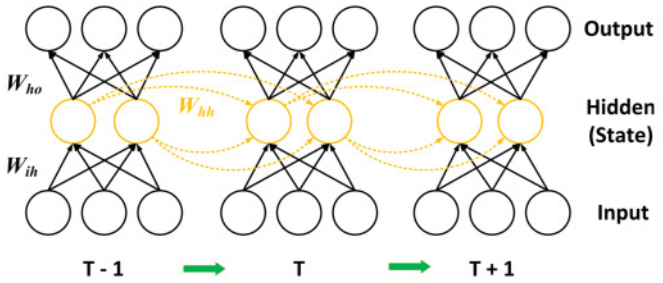
Over the last few years, a large number of DL approaches has been presented. These methods can be divided into two general categories based on how they are used [Deng 2014]: (1) discriminative and (2) generative. Discriminative methods directly evaluate the probability of an output given a certain input, while generative methods estimate the input-output joint probability distribution. Two very popular discriminative architectures are CNNs and RNNs, while DBNs and AEs are two indicative examples of generative models. In the following, the main concepts and recent developments regarding the aforementioned architectures are briefly described. Furthermore, details about training DNNs and optimization techniques proposed so far for enhancing their performance and efficiency are discussed in the following two subsections.

3.1.1. CNNs. CNNs [LeCun et al. 1998] are an example of feedforward networks that have proven to perform remarkably well in several tasks. As implied by their name, Convolutional Networks employ the convolution operation instead of plain matrix multiplication applied in traditional neural networks. A typical convolutional network (illustrated in Figure 2(a)) consists of a combination of three main layer types: convolutional layers, pooling (or subsampling) layers, and fully connected (FC) layers. A number of *filters* are used in order to convolve the input image or previous layer's output. Then, the output values of this operation pass through a nonlinear activation function (also called *nonlinearity*) and afterward, some form of pooling is applied resulting in an equal number of *feature maps* that are given as input to the next layer. On top of the convolutional and pooling layers stack, one or more FC layers are added. In recognition/classification tasks, the last FC layer is usually connected to a classifier (e.g., softmax is a linear classifier commonly used) that outputs the network's response to the initial input data. Each convolutional or FC layer is related to specific parameters/weights that have to be learned. The number of the parameters in each layer is directly related to the number and size of the applied filters.

One important property of Convolutional Networks is *parameter/weight sharing* (also referred to as weight replication). It means that every filter used in a convolutional layer is applied to the whole image (with the same weights) and not just in one spatial location as is happening in traditional NNs. This leads to an important reduction of the model's storage requirements and in addition makes the layer invariant to translation. Each filter is a rectangular matrix of size $p \times p$ applied to small, localized regions of the previous layer's feature maps. This $p \times p$ region is termed as the filter's *receptive field*. In order for the filter to be applied, its weight matrix has to be moved across the input



(a) A typical CNN architecture [LeCun et al. 2010]



(b) RNN architecture [Li et al. 2014b]

Fig. 2. Convolutional and Recurrent Neural Networks Architectures.

image or a layer's feature map in both horizontal and vertical direction. The number of pixels that the top left corner of the matrix will be moved (right or down) is called *stride*. If a large stride is used, the filter will be applied less times leading to a smaller output size and vice versa. Given an input of size $I \times I$ and a filter of size $p \times p$ with stride s, the formula for computing the output feature map's size is $(I - p)/s + 1$.

The result of the convolution becomes the input of a fixed mathematical operation defined by an appropriate activation function. In previous years, nonlinearities such as the sigmoid or hyperbolic tangent (i.e., tanh) were commonly used in neural networks, but after the experimental results of several works like Jarrett et al. [2009], Nair and Hinton [2010], and Krizhevsky et al. [2012], Rectified Linear Units (ReLUs) are currently widely adopted since they have been proven faster to train. The standard ReLU function has the form of $rect(x) = max(0, x)$, that is, simply thresholds the activation to zero. Variants of the standard approach, for example, the Leaky ReLU [Maas et al. 2013] that outputs a small negative slope (instead of zero) when given negative input, the Parametric ReLU [He et al. 2015b] and the Randomized Leaky ReLU have also been proposed. Xu et al. [2015b] evaluated the four Rectified Units in the task of image classification and concluded that for small datasets the variants of the standard ReLU perform better, while He et al. proposed Parametric ReLU and reported remarkable results in image classification surpassing for the first time the human-level performance on the ImageNet 2012 dataset. More complex activation functions have also been proposed. Maxout units [Goodfellow et al. 2013], for example, are a new type of activation function that at first divides the inputs of the activation function into k groups. Then, each maxout unit takes the maximum over one of these k groups. The reported experimental evaluation demonstrated maxout's robustness and its excellent performance especially when used in combination with dropout [Hinton et al. 2012].

In the typical architecture of a CNN, pooling layers usually follow the convolutional layers. Pooling is used for reducing the spatial size of the feature maps and hence, the

computational cost while preserving all the important information. At the same time, pooling makes the representation invariant to small translations and rotation. The most popular pooling function is the max pooling that returns the maximum value of the receptive field. Other notable pooling functions are the average-pooling function and the subsampling function (which is a form of average pooling) according to which each unit computes the average of a local neighborhood, then multiplies it with a trainable coefficient and adds a trainable bias. Finally, it passes the result through a sigmoid function. This kind of pooling was used in the LeNet-5 model of LeCun et al. [1998] designed for handwriting recognition. Experimental evaluation of the max-pooling and subsampling functions was conducted in Scherer et al. [2010] showing the former's superiority, while a theoretical analysis of average and max pooling was presented in Boureau et al. [2010]. More sophisticated pooling methods, for example, stochastic pooling [Zeiler and Fergus 2013], spatial pyramid pooling [He et al. 2014], and def-pooling [Ouyang et al. 2014], have been recently proposed in order to overcome problems like overfitting, fixed-size input image, and deformation.

The last layers of a CNN are FC layers, i.e., layers where each of their units takes as input the output of all units of the previous layer. The first FC layer of a network transforms the output of the last convolutional (or pooling) layer into a vector of predefined length. This vector can either be provided as input to a classifier placed on top of the FC layer (e.g., softmax) or can be considered as a feature vector in order to tackle a different task. Two or three FC layers are typically employed in a deep network. FC layers perform linear multiplication of the input with the weight matrix and contain the highest number of parameters in a network, therefore training them is computationally expensive. Szegedy et al. [2014] tackled this problem in their GoogLeNet architecture (winner of the ILSVRC 2014 challenge) by substituting the FC layers with sparsely connected ones. A different deep network structure termed as "Network in Network" (NIN) was presented by Lin et al. [2013] reporting state-of-the-art results in the classification task for two out of four employed datasets and comparable with the state-of-the-art results for the remaining two datasets. In the NIN structure, convolutional layers were substituted by micronetworks, that is, multiple FC layers with nonlinear activation functions [Lin et al. 2013], which composed the so-called *mlpconv* layers. In addition, the top FC layers of a standard CNN were replaced by global average pooling: in the last *mlpconv* layer, separate feature maps were created for each category, then the average of each map was computed and the resulting vector was fed into the softmax layer for the final classification.

3.1.2. RNNs. A RNN is a powerful architecture typically used to model sequential data, such as text (e.g., see Sutskever et al. [2011]) and sound (e.g., see Graves et al. [2013]). Every RNN is parametrized by three weight matrices (input-to-hidden, hidden-to-hidden, and hidden-to-output) and three bias vectors (hidden, output, and the initial bias vector). Given an input and the desired output, a RNN updates iteratively over time its hidden state through some nonlinearity (e.g., the hyperbolic tangent or the sigmoid) and then, makes a prediction of its output.

More specifically, the hidden state of the network is computed at each timestep based on three values: (1) the input data at this timestep multiplied by the input-to-hidden matrix, (2) the hidden state of the previous timestep multiplied by the hidden-to-hidden weight matrix, and (3) the bias of the hidden layer. Correspondingly, the output of the network in a specific timestep is evaluated based on the output layer's bias and the state of the hidden layer at this timestep multiplied by the hidden-to-output weight matrix. The connections between the input, hidden, and output layers through time are depicted in Figure 2(b). Since the same task is performed in each timestep (just with different input data each time), the weight matrices of a RNN are shared across

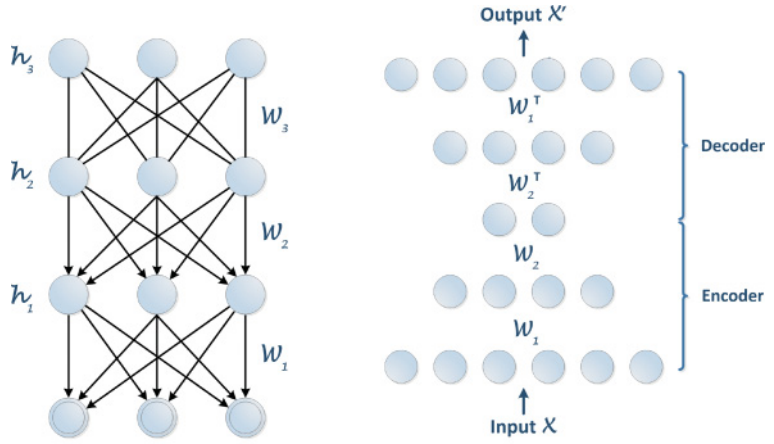


Fig. 3. Deep Belief Network and Autoencoder Architectures.

all timesteps resulting in a significantly reduced number of parameters in comparison to a typical DNN.

The RNNs' power is derived from their high-dimensional hidden state that integrates new information in each iteration and enables them to accurately compute the output. However, a RNN's architecture differs considerably from a CNN and its adaptation as a "deep" model is not straightforward. Pascanu et al. [2013a] elaborated on the depth of a RNN and proposed ways of building deep RNNs. In fact, they proposed two novel architectures of deep RNNs and evaluated them on two tasks, that is, polyphonic music prediction and language modeling, demonstrating their dynamic.

Interestingly, applications of RNNs on the image domain (specifically, for the task of image modeling/generation) have recently been published reporting promising results [Theis and Bethge 2015; Gregor et al. 2015]. Furthermore, an alternative DNN architecture for object recognition, called ReNet, was presented in Visin et al. [2015]. In ReNet, all convolutional layers were substituted with four RNNs, two of which swept the image horizontally (row by row) and the other two vertically (column by column) in both directions. These spatially recurrent layers ensure that the output activation in a specific location is computed with respect to the whole image and not just a local region. Experimental evaluation indicated that the ReNet model can achieve comparable performance to CNNs. This indication was later confirmed by other works as well (e.g., Byeon et al. [2015] and Yan et al. [2016] successfully used spatial RNNs for semantic segmentation, while Bell et al. [2015] exploited them for object detection).

3.1.3. Deep Belief Networks. Deep Belief Networks (DBNs) are probabilistic generative models consisting of multiple layers of stochastic hidden variables [Deng 2014]. All layers in a DBN interact with directed connections except for the top two, which form an undirected bipartite graph. Units belonging to the same layer (either visible or hidden) are not connected. An example DBN with three hidden layers is depicted in Figure 3(a). Every layer identifies correlations among the units of the layer beneath.

DBNs are the first effective DL method introduced by Hinton et al. [2006]. In this work, Hinton et al. [2006] trained successfully a DBN and used it to tackle the task of handwritten digit recognition. Until then, learning in densely connected, directed belief nets with multiple hidden layers was very difficult. Hinton et al. [2006] though, proposed a layer-by-layer, unsupervised method to train such a model by stacking RBMs (i.e., two-layer networks with one visible and one hidden layer) on top of each

other. The presented algorithm was both fast and efficient. Shortly after, the same model was further studied and extended by Bengio et al. [2007] gaining considerable popularity. Since then, DBNs have been used, amongst others, for object recognition, acoustic modeling, and speech recognition.

3.1.4. AEs. AEs [Hinton and Salakhutdinov 2006] are a different model relying on unsupervised training originally used for dimensionality reduction [Rumelhart et al. 1986]. The typical AE model, shown in Figure 3(b), includes two parts: the *encoder* and the *decoder*. The encoder maps the input data to a hidden representation/code using a nonlinear activation function (such as the logistic sigmoid), a weight matrix, and a bias. Then, the decoder maps back the hidden representation resulting in a reconstructed version of the input data. The most appropriate parameters, that is, the ones that minimize the reconstruction error, are determined by training. The cross-entropy loss function is utilized for evaluating the reconstruction error usually when the sigmoid function is used by the decoder, while the squared-error is commonly used when the decoder employs the identity function. In order to facilitate the training process, *tied weights* are often adopted, that is, the transpose of the encoder's weight matrix is used as the decoder's weight matrix.

A variant of the traditional AE model was proposed in Ranzato et al. [2008] in which the possibility of using higher-dimensional (compared to input) but sparse representations was explored. Sparse overcomplete representations were proven to work well in practice. Vincent et al. [2010] proposed a form of regularization by using Denoising Autoencoders (DAEs). A DAE accepts as input a corrupted version of the original input data (e.g., after adding isotropic Gaussian noise) and outputs its clean version. The authors argued that the DAEs were able to extract more useful higher level representations than the standard AEs. The Contractive Autoencoder (CAE) is an alternative regularization technique presented in Rifai et al. [2011]. Targeting at invariance to small variations of the input data, a CAE minimizes the squared Frobenius norm of the Jacobian matrix of the nonlinear mapping between the input and the hidden representation [Rifai et al. 2011]. Experimental evaluation suggested that CAEs perform better than the standard or denoising autoencoders.

3.2. Training DNNs

The first deep architectures proposed and successfully used for computer vision tasks [Hinton et al. 2006; Bengio et al. 2007] included an unsupervised pretraining stage for initializing the weights of all layers and a final, global fine-tuning stage based on an objective function determined by the specific task. Until then, deep multilayer networks were trained starting from random weight initializations and usually performed poorly. In addition, their training was an extremely difficult task due to the large amount of computation time required and the risk of overfitting. Hinton's algorithm [Hinton et al. 2006] and following works such as Bengio et al. [2007], Ranzato et al. [2008], and Lee et al. [2008] though, set the basis for the important advancements taking place in the field of DL until this day.

The original pretraining stage proposed by Hinton et al. [2006] treated the network as a stack of RBMs where each of them was trained separately from the others (e.g., using the Contrastive Divergence method [Hinton 2002]) in a sequential order since the learned activations of one RBM became the input data for training the next. Afterward, a final fine-tuning of the whole network was conducted using a variant of Hinton's wake-sleep algorithm [Hinton et al. 1995]. In addition, Hinton and Salakhutdinov [2006] employed pretraining in order to initialize the weights of a deep AE with values that approximated a good solution. Then, the complete AE was fine-tuned using backpropagation [Hinton and Salakhutdinov 2006] in order to minimize the reconstruction

error. Shortly after, a variant of this layerwise unsupervised pretraining was proven to be an efficient method to train deep networks performing better than similar supervised approaches [Bengio et al. 2007].

After pretraining, a global fine-tuning (supervised training) of the entire network is required in order to finalize the weights. During this stage, a cost function is used in order to measure the network's prediction error (i.e., difference between output and target values). Common cost functions (i.e., training criteria) in feedforward networks include the sum of squared errors and the cross-entropy cost function. In order to minimize the error, partial derivatives of the network's parameters need to be computed. Since the middle 1980s, gradient-based learning methods and backpropagation have started to become very popular for approximating the partial derivatives. The standard gradient descent method performs one parameter update using the whole set of training samples [Bengio 2012]. In recent neural network models though, Stochastic (or online) Gradient Descent (SGD) is more commonly used. SGD updates the network's parameters after each training sample is seen [Bengio 2012]. So far, SGD has been used for training CNNs as well as AEs. Momentum methods update the parameters in a direction where the cost function is constantly reduced, significantly accelerating the overall optimization. In some cases, minibatches are used, that is, small groups of training samples, and the average of each group's gradients is computed and used in the parameters update. In order to efficiently train a neural network, a significant number of parameters has to be tuned [Bengio 2012]. The initial learning rate, the momentum, and the minibatch size are a few of them affecting the training procedure. Other parameters related to the adapted model also have an impact on the network's performance. The task of identifying good values for the parameters of a network is known as hyperparameter optimization [Bergstra et al. 2011; Bergstra and Bengio 2012].

Significant advancements have also been made in the field of RNNs training. In the past, RNNs were trained using a generalization of backpropagation for feedforward networks known as "Backpropagation Through Time" (BPTT) [Rumelhart et al. 1986; Werbos 1990]. However, they were very difficult to train [Pascanu et al. 2013b] mainly due to the *Vanishing Gradient Problem* [Hochreiter 1998] initially acknowledged by Hochreiter in his diploma thesis [Hochreiter 1991]. It describes the situation where, during the backpropagation process of a deep network's training, the gradients start to diminish, resulting in very slow learning rates in the lower layers of the network and hence, in poor performance of the overall architecture. This problem appears both in feedforward and recurrent neural networks. An approach for addressing the vanishing gradients in RNNs was presented by Schmidhuber [1992]. In his work, layer-by-layer pretraining in an unsupervised fashion was proposed in order to initialize the weights in the lower layers. It was proven that these pretrained weights require only a small adaptation to lead to satisfactory results. A few years later, Hochreiter and Schmidhuber [1997] additionally proposed Long Short-Term Memory (LSTM) network for tackling the vanishing gradients in RNNs. A different approach for efficiently training RNNs was described in Martens and Sutskever [2011]. In this work, the combination of Hessian-Free (HF) optimization with a structural dumping was proposed in order to avoid the drawbacks of backpropagation. HF optimization can be used for training RNNs from random initializations without any use of pretraining. A thorough study on training RNNs was provided by Sutskever [2012].

One of the most important issues of concern regarding deep networks training is *overfitting*, that is, training a network to fit perfectly the trained data with the risk of performing poorly on unseen data. In order to achieve generalization, many *regularization* techniques have been proposed. Probably the simplest one is *early stopping*. According to this method, the original training set is split into a smaller training set

and a validation set. The network is trained with the new training set and the validation set is used to compute and record the per-example error in certain time intervals. When the validation set's error stops decreasing, the training also stops even if convergence is not yet achieved. *Data augmentation* is another straightforward way of enhancing generalization. It involves the artificial generation of extra training data from the available ones, for example, by cropping, scaling, or rotating images. Another popular technique is *weight decay* [Krogh and Hertz 1992], also known as L_2 *regularization*. Weight decay involves either penalizing the squared weights by adding an additional term in the cost function or putting a constraint on the maximum squared length of the incoming weight vector of each unit. When the L_2 norm surpasses the predefined limit, the weight vector is scaled down. A similar method is L_1 regularization where the sum of the absolute values of the weights (instead of the squared weights) is penalized. *Dropout* [Hinton et al. 2012] is a recently proposed technique widely adopted in modern deep architectures. It includes temporarily removing some randomly chosen input and output units from the FC layers of a network during the training process in order to limit coadaptations between the units and hence, increase generalization. Experimental evaluations showed that the combination of dropout with L_2 normalization further boosts the performance [Hinton et al. 2012]. Wu and Gu [2015] applied dropout to the input of the max-pooling layers of a CNN achieving enhanced performance, and showed experimentally that the simultaneous use of dropout in different layers can be beneficial. Despite the fact that dropout can successfully address overfitting, it comes with the cost of increasing the training time since in practice it includes the training of many random architectures. A thorough mathematical analysis of certain attributes of dropout can be found in Baldi and Sadowski [2013]. A generalization of Dropout, termed as *DropConnect* was proposed in Wan et al. [2013]. DropConnect instead of randomly dropping out units, drops out the weights. In their work, Wan et al. showed that DropConnect yields superior performance in comparison to Dropout when used for image classification. Stochastic pooling [Zeiler and Fergus 2013] is also a method for regularizing large CNNs. At first, probabilities are computed for all the activations within a region by normalizing them. Based on these probabilities, a multinomial distribution is formed and finally used to randomly sample the activation that the pooling operation will return. Stochastic pooling is applied to the convolutional layers and it can be combined with any other technique, such as weight decay and dropout.

3.3. Optimizing DNNs

With constantly enhanced computational power being available due to parallelization and GPU advancements, neural networks become bigger and bigger both in terms of depth (i.e., more layers) and width (i.e., more filters). Deeper networks typically lead to boosted performance; however, the increased computational cost required for training them constitutes a significant bottleneck for exploiting them in real-time applications. At the same time, since more units are added to the proposed models, millions of parameters are involved in their training, hence storage cost also increases. As a result, considerable effort has been devoted toward compressing and speeding up DNNs.

In a typical CNN, the largest amount of processing time is taken by the convolutional layers, hence reducing their computational cost has been the subject of many recent works. In Vanhoucke et al. [2011], a significant speedup was achieved to the execution of CNNs by taking advantage of several modern CPUs properties and hardware-specific optimizations. Mathieu et al. [2013] managed to improve the train and test time of CNNs by using Fast Fourier Transforms (FFTs) for performing the convolutions. The proposed algorithm was developed to run on GPU. The works of Denton et al. [2014], Jaderberg et al. [2014], and Lebedev et al. [2014] accelerate CNNs test time based

on decomposing layers. All the aforementioned techniques are applied to one or only a few number of layers since accelerating whole deep networks is very challenging, especially for complex tasks (e.g., ImageNet classification). A method for accelerating nonlinear convolutional networks that can be applied to the whole network is proposed by Zhang et al. [2014] demonstrating speedup on a large network trained for ImageNet. Considerable effort has also been allocated in optimizing the training cost of RNNs. An efficient GPU implementation of large-scale RNNs was described in Li et al. [2014b]. Notable speedup was achieved using parallelism within each iteration and advanced tools for hardware exploitation.

Except for the computational time, significant work has been done toward reducing the storage cost of DNNs. Neural network compression has mostly been applied to FC layers since these typically include the highest number of parameters in a network. Denil et al. [2013] highlighted the redundancies existing in the parameters of a neural network and showed that most of the parameters of a layer can be predicted from a small subset of them. In Gong et al. [2014], several vector quantization methods were tested for compressing the parameters of densely connected layers of a CNN. It was proven experimentally that structured quantization methods like product quantization [Jegou et al. 2011] perform best while even a simple k-means can lead to satisfying results. The authors reported compressing the network's parameters even up to 24 times without sacrificing accuracy. Chen et al. [2015a] used the Hashing Trick to compress a network's parameters and introduced a novel deep architecture called HashedNets. In their network, a low-cost hash function was utilized for grouping the weights into hash buckets, therefore all parameters belonging to the same bucket had the same weight value. Recently, Hinton et al. [2015] introduced Knowledge Distillation for neural network compression. According to their proposed framework, an ensemble of deep networks (teacher) is initially trained using the original labels and afterward, a smaller distilled network (student) is trained using the combination of the original labels and the output of the teacher network. Experimental evaluation indicated that the (compressed) student network provides better generalization than the teacher. Knowledge transfer has also been explored in Romero et al. [2014] where thin and deep networks, called FitNets, were trained in order to compress wider and shallower networks. Romero et al. [2014] trained the student's network using the hidden layers of the teacher's network instead of the output layer and reported better generalization and reduced computational cost.

4. ADVANCES IN DEEP LEARNING WITH 3D DATA

After their huge success in several 2D computer vision tasks, DNNs have started to become popular in the 3D domain as well, achieving remarkable performance. The complex intrinsic nature of the 3D data, though, transforms the way in which they should be treated and ultimately provided to a DNN into a major challenge. In order to address this issue and tackle tasks such as 3D object recognition and retrieval, researchers followed different directions. We classify the existing approaches into five main categories: (i) The first category includes methods that extract descriptors from the 3D data and give these as input to the DNN. (ii) The approaches belonging to the second category exploit RGB-D data (i.e., separate color and depth channels) captured from popular low-cost cameras like Microsoft's Kinect. (iii) Deep architectures designed to have direct access to the 3D data form the third category. (iv) The fourth category includes methods utilizing one or more 2D projections/views of the 3D object/scene captured from different viewpoints and use them to feed the employed deep model. Finally, (v) DL methods designed for data captured from hyperspectral cameras are included in the last category. In the following, the works of each category are separately described (Sections 4.1–4.5). Moreover, DL systems utilizing multiple 3D data modalities are

discussed in Section 4.6. An overview of all methods is provided at the end of this section (Section 4.7) along with concluding remarks and observations. In order to facilitate the comparison between the examined works, comprehensive tables organized per task are also available (Tables I–III) presenting the key components of each method.

4.1. DL Architectures Exploiting Descriptors Extracted from 3D Data

Low-level descriptors extraction is often part of a recognition or retrieval pipeline. In the 3D domain though, an efficient high-level 3D shape descriptor is usually of demand due to the complexity of the 3D data. A common practice is to extract low-level descriptors and then, provide them as input to a DNN in order to get a more effective high-level representation for recognition, retrieval, or other tasks.

Liu et al. [2014] adopted low-level descriptors extraction in order to represent 3D shapes before feeding them to a DBN. More specifically, 200 depth images were obtained from each 3D model and SIFT [Lowe 1999] descriptors were extracted from each image. Afterward, the BoW paradigm was adopted and every SIFT descriptor extracted from the 3D models was encoded based on the constructed vocabulary. The BoW vectors of the 3D models were provided as input data to the DBN, which was trained in a greedy layerwise fashion. Experiments on both 3D classification and retrieval demonstrated the proposed method's superior performance in comparison to the standard BoW paradigm indicating that the high-level descriptors generated by the deep network are more discriminative. In the work of Bu et al. [2014], two local descriptors, that is, Scale Invariant Heat Kernel Signature (SI-HKS) [Bronstein and Kokkinos 2010] and Average Geodesic Distance (AGD) [Hilaga et al. 2001], were initially extracted from each 3D model and a final low-level descriptor was formed by concatenating the first six frequency components of SI-HKS with the AGD value. Afterward, a variant of the standard BoW (or Bag-of-Features) model, called Geodesics-Aware Bag-of-Features (GA-BoF), which takes under consideration the spatial relationship among every two BoFs, was employed. The middle-level representation obtained from the GA-BoF was given as input to a DBN trained layerwise using contrastive divergence. The high-level descriptors used for similarity search in 3D shape retrieval context were the output of the last layer, while for the 3D model recognition class labels were predicted through supervised learning based on training examples. For their classification experiments, the DBNs were regarded as a dimensionality reduction method and therefore, they were compared to other similar algorithms like Principal Component Analysis (PCA) and Multi-Dimensional Scaling (MDS). In the retrieval task, DBNs were compared to several other descriptors such as Heat Kernel Signature (HKS). Results on three datasets demonstrated that the extracted high-level representation is more discriminative leading to enhanced or equivalent retrieval and recognition performance. The framework proposed by Bu et al. [2014] was later adapted for 3D shape correspondence and symmetry detection as well [Bu et al. 2015]. The high-level descriptor generated by the proposed DBN, referred to as Local Deep Feature (LDF), was proven to achieve notable performance compared to other descriptors, for example, SI-HKS. Moreover, in their extended work, Bu et al. exploited a GPU-based DL toolbox and managed to considerably accelerate the learning process.

The distributions of HKSs of shape at different scales were extracted as low-level descriptors from 3D objects in Xie et al. [2015a]. This multiscale shape distribution was then fed to multiple discriminative AEs that were trained to learn a high-level descriptor representation for 3D shape retrieval. In order to enhance the discriminative power of the generated representation, a Fisher discrimination criterion was also employed. The proposed 3D shape descriptor was finally formed after concatenating the activations of all hidden layers of the AEs. Experiments on 3D shape matching and retrieval revealed the proposed 3D descriptor's robustness and showed that it is

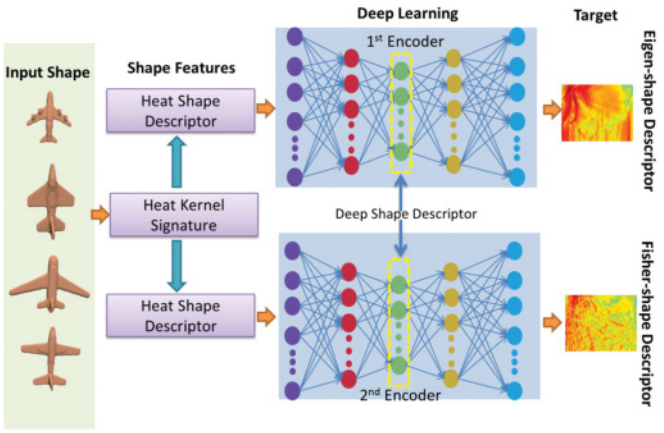


Fig. 4. Learning 3D Deep Shape Descriptor [Fang et al. 2015].

insensitive to deformations. A similar framework was presented in Fang et al. [2015]. The authors proposed the 3D Deep Shape Descriptor, which was generated using the Heat Shape Descriptor (HeatSD), that is, a multiscale shape descriptor based on HKS, and a many-to-one encoder (an architecture that ensures that the exact same output will be generated for the inputs of the same category). PCA and Linear Discriminant Analysis (LDA) were applied on the set of HeatSDs extracted from the 3D models in order to generate the Eigen-Shape Descriptor (ESD) and Fisher-Shape Descriptor (FSD), respectively. Precomputed ESDs and FSDs were used as target values in order to guide the training of two separate encoders, one for each descriptor. The proposed training strategy maximized the interclass margin and at the same time minimized the intraclass variance boosting the discriminative power of the deep shape descriptor. The overall process is depicted in Figure 4. 3D Deep Shape Descriptor achieved good results in the context of 3D shape retrieval.

4.2. DL Architectures Exploiting RGB-D Data

Several RGB-D datasets have became available in the last few years thanks to the increased popularity of RGB-D sensors like Microsoft's Kinect. These sensors provide the extra modality of depth in addition to the color information, and many works utilize these kind of 3D data to tackle tasks such as 3D object recognition, retrieval, or semantic segmentation. A study of data fusion methods for RGB-D visual recognition can be found in Sanchez-Riera et al. [2016].

One of the first approaches for 3D object classification based on RGB-D data was presented by Socher et al. [2012]. The authors proposed a combination of convolutional and recursive neural networks where color and depth channels were processed separately. At first, two single-layer CNNs were employed in order to extract low-level descriptors from the RGB and depth images. Then, each CNN's output was forwarded to a different set of RNNs initialized with random weights. The RNN descriptors extracted from each modality were finally merged and provided to a joint softmax classifier. The proposed method demonstrated accurate performance in classifying household objects. Couprie et al. [2013] used a multiscale CNN for semantic segmentation of indoor RGB-D scenes. The network processed the input depth and RGB images at three different scales and the upsampled results were combined and forwarded to a classifier in order to get object class labels. The final labeling of the scene was obtained by merging the classifier's predictions with a superpixels segmentation of the scene performed in parallel.

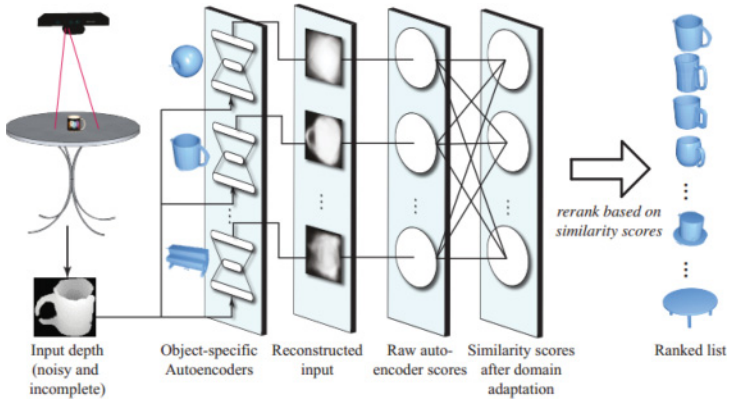


Fig. 5. Depth-based retrieval with an ensemble of Contractive AEs [Feng et al. 2016].

Experimental evaluation showed the introduced method is more efficient and faster than previous approaches. The possibility of using transfer learning between CNNs for object recognition was investigated in Alexandre [2014]. The author proposed the employment of four independent CNNs for processing the four input channels of an RGB-D image. The four CNNs were trained sequentially, passing the weights of a trained CNN as input to the next. Experiments on 3D objects from 10 categories indicated that the proposed training strategy can boost the performance.

In the work of Schwarz et al. [2015], transfer learning from deep CNNs was also explored for addressing RGB-D object recognition. The proposed architecture employed a pretrained CNN for image categorization and fed it with the color and depth channels separately. Preprocessing the input data was required in order to transform them to an appropriate format. Regarding depth information, Schwarz et al. proposed to render objects from a canonical perspective and colorized depth based on the distance from the object center. Experiments were conducted using the output of the last two FC layers of the network as descriptors, while SVMs were ultimately utilized in order to predict the category, instance, and pose of the tested objects. The task of RGB-D object recognition was also addressed by Eitel et al. [2015]. A two-stream CNN architecture for RGB-D object recognition was designed in this work too. Each stream (one for color and the other for depth) contained five convolutional and two FC layers. The two streams were originally trained individually and afterward, they were fused together in a FC layer and a softmax classifier. The two CNNs employed for recognition were pretrained for the task of object classification on the ImageNet dataset hence, preprocessing of the input data (especially depth) was required. A new way to encode depth to color images (similar to the one proposed by Schwarz et al. [2015]) was presented, which, as shown experimentally, outperformed other existing approaches. In addition, a new data augmentation scheme that generated artificial noise patterns and used them as additional training samples was also exploited. Extensive experimental results demonstrated the accurate performance of the proposed approach in the task of recognizing objects in real-world, noisy scenes. In addition, the proposed method achieved higher recognition accuracy compared to other methods such as Socher et al. [2012] and Schwarz et al. [2015]. An ensemble of CAEs (shown in Figure 5) was used by Feng et al. [2016] to tackle the task of 3D shape retrieval using as query only one depth image captured from Kinect-like cameras. Each AE was trained using SDG on a different database CAD model. Due to the different data type of the queries (i.e., depth data) in comparison to the training data (i.e., CAD models), the output scores of all AEs were forwarded to a

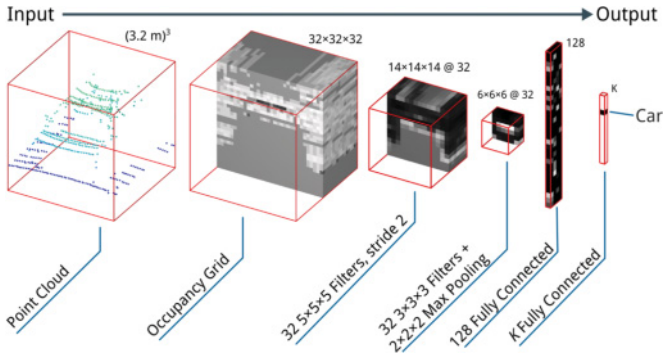


Fig. 6. VoxNet[4] architecture for 3D object recognition [Maturana and Scherer 2015].

novel layer, called Domain Adaptation Layer (DAL), in order to finalize the retrieved ranking list. Experimental evaluation indicated that the proposed approach achieved increased performance compared to other related methods like the combination of Histogram of Oriented Gradients (HOG) descriptor with L_2 norm or a global AE.

4.3. DL Architectures Exploiting Directly 3D Data

During the last couple of years, approaches exploiting the complete 3D geometry of the captured scenes have started to appear. A novel approach toward 3D shape representation exploiting the full 3D structure was presented in Wu et al. [2015]. In the proposed architecture, called *3D ShapeNets*, 3D shapes were provided as input (a 3D voxel grid where each voxel was a binary variable indicating whether it belonged to the 3D shape or it was empty space), while the DBN model was employed. In order to diminish the huge number of parameters required from feeding a fully connected DBN with a 3D voxel volume of normal resolution, convolution with 3D filters was applied. Most specifically, a Convolutional Deep Belief Network (CDBN) with five layers (three convolutional, one fully connected, and one output layer) was proposed. The model was initially pretrained layerwise and afterward, fine-tuned by backpropagation. Standard contrastive divergence was used for training the first four layers, but the more sophisticated Fast Persistent Contrastive Divergence (FPCD) was employed for training the top layer. The proposed framework was tested on the tasks of 3D shape classification and retrieval, next-best view prediction, and view-based 2.5D recognition outperforming other state-of-the-art methods. Moreover, the authors released *ModelNet*, that is, a new, large-scale 3D dataset with CAD models from 662 unique categories.

Maturana and Scherer have also employed volumetric (i.e., spatially 3D) representation of the 3D data to perform 3D object recognition [Maturana and Scherer 2015]. In the proposed *VoxNet* architecture (depicted in Figure 6), a volumetric occupancy grid of size $32 \times 32 \times 32$ voxels was at first generated from a point cloud's segment that was then given as input to a CNN. The employed network was constructed using two convolutional (with 3D filters), one pooling, and two FC layers, while it was trained using SGD with momentum. An object class label was finally predicted for each segment. Data from three different domains, that is, LIDAR data point clouds, RGB-D point clouds, and CAD models, were used for evaluating VoxNet. Experiments showed that the proposed architecture resulted in better classification accuracy than similar approaches while demonstrating real-time performance. More specifically, VoxNet outperformed 3D ShapeNets [Wu et al. 2015] in the classification task using ModelNet10

[4]Image from http://www.dimatura.net/pages/3dcnn.html.

and ModelNet40 as well as NYU v2 dataset [Silberman et al. 2012] when training from scratch. On the other hand, 3D ShapeNets performed better on the NYU v2 dataset when using the models pretrained for ModelNet10. Sedaghat et al. [2016] modified VoxNet's architecture in such a way that the object's orientation was taken into account. In their final model, the class labels were extracted directly from the orientation activations. Improved classification results were reported for ModelNet10 and other datasets.

A new 3D descriptor learning method combining the strengths of CNNs, AEs, and ELMs, that is, *Convolutional AutoEncoder Extreme Learning Machine* (CAE-ELM), was recently proposed in Wang et al. [2016]. *Extreme Learning Machines* (ELMs) are feedforward networks containing only one hidden layer proposed in Huang et al. [2006]. The hidden layer of an ELM is actually fixed and random, while the output layer is trained in a supervised manner according to the task of interest. A deep extension of the standard ELM model with multiple hidden layers was proposed by Kasun et al. [2013]. The architecture proposed in Wang et al. [2016] contained the following components: (i) Convolutional feature map generation: in this part of the network, the 3D input data, that is, voxel and Signed Distance Field (SDF) data, were convolved with randomly generated 3D kernels and convolutional feature maps were computed. Following, average pooling was applied to the feature maps in order to maintain rotation invariance. (ii) AE descriptors extraction: after pooling, each feature map was provided as input to a separate AE. All AEs were originally initialized with random weights and their final (output) weights were learned via training. (iii) ELM classifier: in the last part of the network, all descriptors extracted from the AEs were concatenated into a vector that was used for predicting the current 3D shape's label. CAE-ELM was tested on ModelNet in the context of 3D shape classification, 3D shape retrieval, and 3D shape completion leading to superior results compared to methods like Wu et al. [2015] and Xie et al. [2015b]. The final architecture used in the experiments included two parallel CAE-ELM layers, one working on the voxel and the other on the SDF data. Han et al. [2016] proposed *Mesh Convolutional Restricted Boltzmann Machines* (MCRBMs) for learning high discriminative 3D features from 3D meshes. The learned features were designed to preserve the structure between local regions and can be used as local or global features. A novel raw representation of the local region, called Local Function Energy Distribution (LFED), was provided as input to the network. In addition, Multiple MCRBMs were combined forming a deeper model, named *Mesh Convolutional Deep Belief Network* (MCDBN). Both of the proposed deep models were tested in the context of global and partial shape retrieval and shape correspondence surpassing the performance of state-of-the-art methods such as BoW, Chen et al. [2003], Kazhdan et al. [2003], Bronstein and Kokkinos [2010], and Wu et al. [2015].

In their seminal work, Qi et al. [2016] elaborated on two factors, that is, network architecture and volume resolution, that affect the performance of volumetric CNNs and proposed two new CNN architectures that improved the current state-of-the-art performance in object classification, that is, VoxNet's [Maturana and Scherer 2015] 83% average class accuracy. The first proposed CNN included a 3D extension of the *mlpconv* layers proposed in Lin et al. [2013] and attempted to emphasize details of the 3D objects by including additional learning tasks classifying parts of an object. This network achieved 86% average class accuracy. The second CNN initially took advantage of long anisotropic kernels to consider long-distance interactions and exploited an adapted NIN network [Lin et al. 2013] for classification resulting in 85.6% average class accuracy. In both networks, training data augmentation by applying several azimuth and elevation rotations was also used along with multiorientation pooling leading to further improved performance. The extensive experimental evaluation highlighted the significance of 3D resolution in volumetric CNNs. Recently, Brock

et al. [2016] presented voxel-based (i.e., fully 3D) models for shape modeling and 3D object classification and reported improved by a large margin classification results on ModelNet10 and ModelNet40 compared to any other 3D or multiview DL approach proposed so far. The authors took advantage of recent advancements in the field of DNNs and designed an architecture that relied on (i) inception-style modules [Szegedy et al. 2016], (ii) batch normalization [Ioffe and Szegedy 2015], (iii) residual connections with preactivation (He et al. [2015a, 2016]) and, (iv) stochastic network depth [Huang et al. 2016]. The proposed model, *Voxception-ResNet* (VRN), is 45 layers deep. The authors reported state-of-the-art classification results for ModelNet40 and ModelNet10 using an ensemble of VRN models. It should be noted that significant data augmentation was required for training such a deep model.

A pipeline for 3D object detection and recognition in RGB-D scenes, referred to as *Deep Sliding Shapes,* was presented in Song and Xiao [2016]. Interestingly, instead of just working on the depth channel, Song and Xiao exploited the raw 3D information of the scenes by converting each depth image to a full 3D voxel grid using a directional Truncated Signed Distance Function (TSDF). A fully 3D convolutional network, called 3D Region Proposal Network (RPN), was then utilized in order to generate 3D object bounding boxes from the 3D voxel grid at two different scales so that it could handle different object sizes. Objectness scores were also provided for each generated object proposal. Moreover, each detected 3D proposal box and its corresponding 2D color patch (i.e., 2D projection of the 3D proposal) were fed to a 3D ConvNet and a 2D ConvNet, respectively, for jointly learning the object's category and 3D box regression. For the object proposal evaluation, the proposed approach was shown to outperform the 3D version of the state-of-the-art Selective Search method [Uijlings et al. 2013], while for the task of object detection Deep Sliding Shapes [Song and Xiao 2016] were compared with their "nondeep" version, that is, 3D Sliding Shapes with handcrafted descriptors [Song and Xiao 2014], and the state-of-the-art 2D Depth-RCNN with ConvNets descriptors [Gupta et al. 2014] resulting in superior performance.

4.4. DL Architectures Exploiting 2D Projections/Views of 3D Objects

Collecting multiple 2D projections rendered from different directions in order to represent a 3D shape/object is a "trick" commonly adopted for 3D shape analysis and understanding. One of the first approaches toward the construction of a DL representation for 3D shapes by projecting them into the 2D space can be found in Zhu et al. [2014]. In the aforementioned work, an AE was used in order to generate a global deep representation of a 3D shape for the application scenario of 3D shape retrieval. Pose normalization for differences in translation and scale was initially applied to each 3D model, while a set of 2D projections was subsequently collected for each of them. After pretraining the stacked RBMs with the projections, the AE was fine-tuned using backpropagation in order to minimize the reconstruction error. Finally, the hidden (code) layer was used for representing the corresponding projection/view of the 3D shape in the retrieval process. Since more than one code was generated for each model (one per projection), a variant of the Hausdorff distance was used to compute the distance between the final representations of two different 3D shapes. Experiments on two popular datasets, that is, Princeton Shape Benchmark (PSB) [Shilane et al. 2004] and Engineering Shape Benchmark (ESB) [Jayanti et al. 2006], indicated that the proposed architecture performed better compared to other global descriptors-based methods (e.g., LFD [Chen et al. 2003]). In addition, the global representation was linearly combined with a local one, that is, the BoW based on SIFT, yielding in boosted performance.

An AE was also adopted for 3D object retrieval in Leng et al. [2015a]. In this method, an extension of the standard AE inspired from CNNs, called *Stacked Local Convolutional AutoEncoder* (SLCAE), was proposed. A Local Convolutional Autoencoder

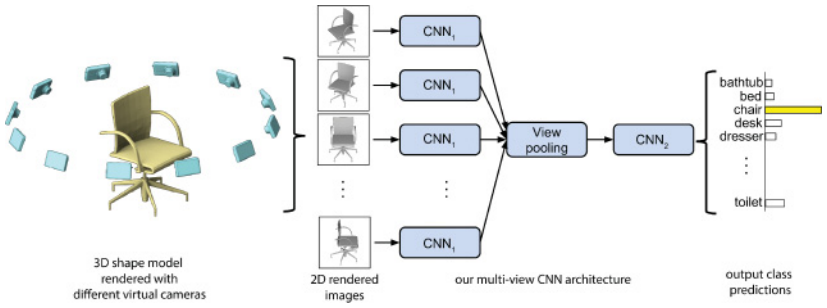


Fig. 7. MVCNN for 3D object classification and retrieval [Su et al. 2015].

(LCAE) is constructed by substituting the FC layers of a standard AE with locally connected layers using the convolution operation. In the stacked version of LCAE, many encoders were placed on top of each other and the output of the last one was used as the representation of a 3D object. The input provided to the proposed AE was multiple depth images of several views of the 3D object, while each layer of the architecture was trained using the gradient descent method. The SLCAE model was compared to other state-of-the-art approaches (e.g., BoW) on three 3D object datasets, that is, PSB [Shilane et al. 2004], National Taiwan University (NTU) dataset [Chen et al. 2003], and SHREC'09 [Godil et al. 2009], leading to superior results. A different architecture, named 3D Convolutional Neural Network (3DCNN), was proposed by Leng et al. [2015b] for dealing with multiple 2D views of a 3D object at the same time. Each object's views were sorted into three reasonable sequences before being fed to the network, so that the views were listed in a fixed order. The 3DCNN was comprised of four convolutional layers, three subsampling layers, and two FC layers. Convolutional layers were initially pretrained in the same way of training an AE. Afterward, the whole network was fine-tuned using backpropagation. The output of the first FC layer was used as the representation of the input data for the retrieval. Evaluation on three datasets (i.e., PSB, NTU, SHREC'09) demonstrated the proposed method's remarkable performance in comparison to other state-of-the-art methods. Despite the good results achieved by the 3DCNN representation though, the reported performance is lower compared to the retrieval performance obtained using the SLCAE [Leng et al. 2015a] on all three employed datasets, which indicates that the latter representation is probably a better choice for this task.

Multiple views of a 3D object were also exploited in the work of Su et al. [2015] in order to build a compact shape descriptor for the tasks of 3D object classification and retrieval. The novel *Multi-View CNN* (MVCNN) architecture, shown in Figure 7, learns to combine any number of input views of an object without any particular order through a view pooling layer. In order to obtain different views of the models, two setups were tested. The first setup included 12 rendered views of the 3D objects by placing an equal number of virtual cameras around them, while the second involved 80 views. All the available views of an object passed through the first part of the network separately and then, elementwise max pooling was performed across all views in the view pooling layer. Finally, the aggregated result passed through the remaining network. For retrieval, the penultimate seventh layer of the network (which is fully connected) was used as shape descriptor. The employed network was pretrained using the ImageNet1K dataset and then, fine-tuned using the 3D dataset ModelNet40 [Wu et al. 2015] that was used in the experimental evaluation of the MVCNN architecture. Reported results on shape classification and retrieval showed that MVCNN outperformed all the other tested methods. It is noteworthy that the proposed shape descriptors surpassed the

performance of the state-of-the-art 3D ShapeNets of Wu et al. [2015] by a large margin, especially in the retrieval task using ModelNet40 dataset. A different approach for exploiting the multiple views of a 3D object was followed by Johns et al. [2016] for the application scenario of multiview object recognition under unconstrained camera trajectories. In this work, the collection of views was organized in pairs that were provided to a CNN together with their relative pose. The VGG-M network [Chatfield et al. 2014] was employed in this case consisting of five convolutional and three FC layers. Grayscale images, depth images, or both can be given as input to the network. The outputs of the convolutional layers from the two images were concatenated before being provided to the first FC layer. The introduced model surpassed the performance of voxel-based 3D ShapeNets [Wu et al. 2015] and the MVCNN approach of Su et al. [2015] on the ModelNet dataset.

A real-time 3D shape search engine based on 2D views of 3D objects was presented in Bai et al. [2016]. The proposed system, named GIFT, exploited GPU for CNN-based feature extraction and utilized two inverted files, that is, one for accelerating the multiview matching process and the other for reranking the initial results. The retrieval process for a query shape was reported to be completed within a second. The authors tested their engine on ModelNet, SHREC14LSGTB [Li et al. 2015], PSB, McGill [Siddiqi et al. 2008], and SHREC'07 watertight models [Giorgi et al. 2007] datasets demonstrating GIFT's superior performance in comparison to state-of-the-art methods (e.g., Chen et al. [2003], Kazhdan et al. [2003], Wu et al. [2015], and Su et al. [2015]). A different approach where 3D models were retrieved based on 2D sketches and 2D views has recently been presented in Wang et al. [2015]. More specifically, Wang et al. proposed an architecture that takes as input a {2D view + sketch} pair of an object. The model consisted of two Siamese CNNs (i.e., two identical subconvolutional networks), one for dealing with the 2D sketch of the 3D object to be retrieved and the other with the 2D view. The two subnetworks were trained separately using SGD and backpropagation. Each subnetwork contained three convolutional layers, each followed by a max-pooling layer, and one FC plus one output layer on top. Every 3D model was characterized by two randomly generated views as far as their angles differed more than 45°. The proposed network was tested on three datasets and achieved the best performance.

In Leng et al. [2014], view-based depth images were also used as input and high-level abstract descriptors were extracted through a DBN to be used for 3D object classification. The employed DBN was trained in an unsupervised layerwise manner using contrastive divergence. After getting the final representation for each model from the trained DBN, a semisupervised learning approach was applied in order to classify each object. Experimental evaluation based on accuracy and Mean Classification Precision (MCP) indicated that the descriptors extracted from the proposed DBN led to considerably better results compared to two composite descriptors that combined several 2D descriptors extracted from different views, that is, the CMVD [Daras and Axenopoulos 2010] with graph learning (CMVD-GL) and Equal Weight using Graph Fusion (EW-GF). Xie et al. [2015b] presented the *Multi-View Deep Extreme Learning Machine* (MVD-ELM) and tested it on the tasks of 3D shape classification and segmentation. Each 3D shape was represented by a collection of 20 2.5D depth images/projections captured uniformly using a sphere centered at each object. The MVD-ELM model contained convolutional and pooling layers. The weights in each convolutional layer were shared across all views. The output weights were optimized based on the extracted feature maps. A Fully Convolutional extension of the proposed model (FC-MVD-ELM) was also presented for the task of 3D shape segmentation. This network contained only two convolutional layers without any pooling layer. FC-MVD-ELM was trained using the multiview depth images of the training examples. Then, all the predicted

labels were projected back into the original 3D mesh. Finally, the segmentation result was smoothed using graph cuts optimization. The proposed models outperformed other relevant approaches (e.g., Wu et al. [2015]) and significantly reduced the training time.

Driven by the indication that multiview DNNs lead to superior performance compared to the ones exploiting the full 3D information of 3D shapes, a recent study and comparison of volumetric versus multiview CNNs for object classification was presented in Qi et al. [2016]. For the case of multiview CNNs, Qi et al. proposed *sphere rendering*, that is, multiresolution 3D filtering in order to exploit information from multiple scales, and in combination with training data augmentation managed to achieve enhancements to the already high performance of MVCNNs on ModelNet40.

4.5. DL Architectures Exploiting HyperSpectral Data

Advanced remote sensing technology is currently being used in several applications. Airborne or spaceborne sensors are often used for capturing HyperSpectral (HS) images typically in the form of a stack of N images of size $[p1 \times p2]$ pixels representing the radiance in the respective band (also referred to as 3D hypercubes) [Bioucas-Dias et al. 2013]. Traditional methods for HS data analysis exploit only the provided spectral information, but works dealing with both spectral and spatial data have also been proposed. Recently, DL methods applied on HS data have started to appear showing promising results. A detailed technical tutorial on state-of-the-art DL methods addressing image preprocessing, pixel-based classification, target recognition, and scene understanding using remote sensing data can be found in Zhang et al. [2016a]. Amongst the referred tasks, pixel-based classification is probably the most popular one.

Toward deep hyperspectral data classification, Hu et al. [2015] employed a CNN to perform the classification directly in the spectral domain. The proposed architecture consisted of an input layer (accepting a vector with each pixel's spectrum), a convolutional layer, a max-pooling layer, a FC layer, and the output layer. The introduced method led to better results compared to SVM-based classifiers and other CNN architectures. In Chen et al. [2014], Stacked AEs (SAEs) were utilized for extracting deep features from the hyperspectral data in order to perform classification. The authors experimented with two frameworks: the first one utilized either spectral features or spatial-dominated features, while the second performed classification based on joint spectral and spatial information. A vector with a pixel's spectrum was provided as input to the AEs for spectral feature extraction, while a $[n \times n]$ neighbor region of a certain pixel was extracted from the original image and was provided as input to the deep model for spatial-dominated feature extraction. The experimental results demonstrated that the deep models performed better than the standard SVM classifiers and that the joint spectral-spatial version of their deep framework outperformed the one using only spectral or spatial information. This observation has also been confirmed by similar frameworks proposed in Chen et al. [2015b, 2016] utilizing DBNs and 3D CNNs, respectively. Makantasis et al. [2015] also proposed a DL-based classification method, which extracts a feature representation from spectral and spatial data based on a CNN. The designed network included two convolutional layers, while a Multilayer Perceptron was employed for the classification. The introduced method was compared to SVM-based classifiers and outperformed them.

A hierarchical model, called *Spectral-Spatial Network* (SSN), was introduced in Zhou and Wei [2016] for addressing HS image classification. The network consisted of several stacked Spectral-Spatial Feature Learning Units (SSFLUs) and a Kernel-based Extreme Learning Machine (KELM) [Huang et al. 2012] on top for performing the classification. Each SSFLU included a LDA step for extracting discriminative spectral features, and the application of Adaptive Weighted Filters (AWFs) for exploiting the spatial information. Extensive experiments demonstrated the SSN's good accuracy and

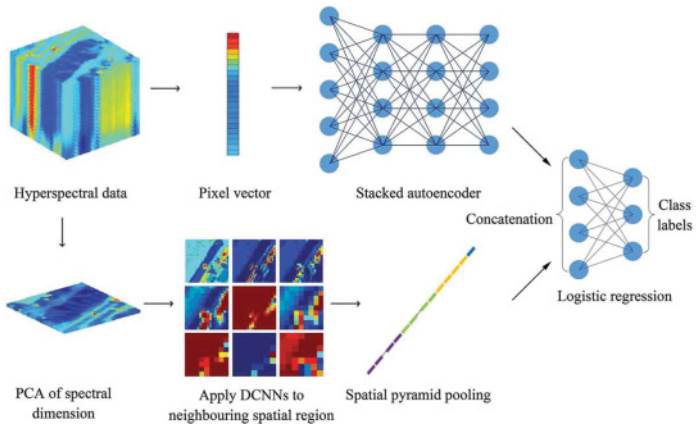


Fig. 8. Joint spectral-spatial classification framework of Yue et al. [2016].

robustness when compared to other state-of-the-art methods. A *Multiscale CNN* was used in the approach of Zhao and Du [2016]. The proposed method exploited first PCA for dimensionality reduction and then, feature extraction was applied in the three PC bands. An image pyramid was constructed for each selected PC band to capture spatial features in multiple scales. Spatial and spectral features were combined to classify each PC band using a Logistic Regression (LR) classifier, while a voting scheme was employed to fuse the classification result of all bands. The authors compared the proposed method to Extended Morphological Profiles (EMPs) [Benediktsson et al. 2005] and composite kernel SVM [Fauvel et al. 2012] indicating its effectiveness. In the work of Yue et al. [2016], Spatial Pyramid Pooling (SPP) was employed for HS image classification. The DL framework, shown in Figure 8, included feature extraction combining the output of Stacked AEs and CNNs and a softmax classifier on top. SPP was applied after the last convolutional layer of the deep CNN allowing the generation of fixed-length features regardless of the scale of the input features. Experimental evaluation showed the proposed method's superior performance compared to RBF-SVM and EMP-SVM.

4.6. DL Architectures Fusing Different 3D Data Modalities

Beyond the approaches presented in the previous subsections, many researchers experimented with more sophisticated methodologies targeting to better results. Motivated by the current trend in 1D/2D DL methods involving the fusion of different data modalities (e.g. Doulamis and Doulamis [2012], Martínez and Yannakakis [2014], Xu et al. [2015a], and Zhang et al. [2016b]), several DL fusion methods have been proposed for 3D data as well. Apart from RGB-D, which has been discussed in depth in Section 4.2 as a special case of data fusion, the fusion of other data modalities has also been attempted. Merentitis and Debes [2015] extracted deep features from hyperspectral and LIDAR data and combined their learned first-layer representations to perform classification using random forests. In the work of Hegde and Zadeh [2016], a fusion of volumetric (i.e., 3D) and pixel (i.e., 2D views) representations was attempted for 3D object classification. More specifically, the authors used AlexNet network [Krizhevsky et al. 2012] for the 2D views of each 3D object, while they proposed two 3D CNNs for the volumetric data. The multiview network performed better on ModelNet40 than the volumetric ones, but the highest performance was achieved by the combination of the three different networks, named *FusionNet*. In a similar vein, RGB, Depth, and Point Cloud data were combined in Zaki et al. [2016]. Depth maps and point

cloud embedding was initially performed, while a CNN pretrained on RGB images was employed for feature extraction. A Hypercube Pyramid descriptor was proposed for representing multiscale, spatially relevant information for object and instance classification using ELMs. The extracted descriptor was fused with the activations of the pretrained network's FC layers creating an even more compact representation. The proposed approach was compared to state-of-the-art methods on two benchmark RGB-D datasets presenting superior performance in terms of recognition accuracy.

4.7. Overview of the DL Architectures Designed for 3D Data

During the last few years, DL methods have been designed and successfully applied to 1D and 2D data. Employing them in the 3D domain though is not as straightforward, since the typical DL architectures are designed to take as input either 1D or 2D data. In order to overcome this barrier, several approaches have been proposed. In this section, five categories were identified and separately analyzed regarding the way in which either the 3D data are manipulated before being provided to a DL architecture or the DL architecture is modified to take directly 3D data as input. In order to deal with the 3D data, many researchers took advantage of the developments in feature engineering. Low-level feature extraction has been used in several computer vision tasks with great success and a large variety of local or global descriptors has been proposed for 3D data so far. Since low-level descriptors are usually not sufficient to characterize the high-level semantics of the 3D objects, the works in this category exploited them in combination with a deep model in order to extract high-level descriptors. However, considering the complexity of 3D data, this representation may be lacking discriminative power since the shallow representations may omit significant information from the 3D representation. RGB-D sensors provide the extra depth modality (in addition to the standard RGB channels) that contains important information about a 3D object's shape. Most researchers dealt with color and depth channels (i.e., images) separately, while others used only the depth information in order to design their systems. The big advantage of these sensors is that they are inexpensive for an average user and at the same time many open-source software solutions exist facilitating their usage. However, their low cost is often combined with noisy and incomplete captured data that probably makes them unsuitable for complex scenarios. Exploiting directly the 3D information by replacing the 2D convolutional layers with 3D ones has been attempted by some recent works. 3D volumetric models provide a rich and powerful representation of 3D shapes including all the important details. Despite the huge advancements made in computational hardware though, their processing is still demanding both in terms of memory and computation time. As a result, low resolutions have only been utilized so far. Other researchers approached the problem from a different angle and utilized one or more 2D views of a scene captured from different viewpoints (multiview). By doing so, the problem is indirectly transformed to the image domain, therefore multiview-based methods can take advantage of the latest advancements in image processing and are straightforward to employ. However, several concerns arise from their exploitation: (1) the full 3D geometry information of a 3D shape is lost in 2D views, and (2) the number of views that should be acquired and the way in which they should be linked for representing a 3D shape is a critical step that could influence both the efficiency and the effectiveness of a proposed method.

In order to understand the relation and also facilitate the comparison between different methods, the works described in Sections 4.1–4.5 addressing 3D object retrieval, 3D object classification/recognition, and 3D hyperspectral classification are separately presented in Tables I, II, and III, respectively. All three tables summarize for each cited work the type of the input data and the employed DL-architecture along with important experimental details, that is, an indicative dataset and evaluation measure

Table I. Deep Learning Architectures for 3D Object Retrieval

			Experimental			
Method	**Input**	**Deep Model**	**Dataset**	**Measure**	**Score**	**Other Methods**
Leng et al. [2015a]	2D views	SLCAE	PSB	FT	67.2%	LFD: 38%
Leng et al. [2015b]	2D views	3D CNN	PSB	FT	63.9%	CMVD: 28.6%
Zhu et al. [2014]	2D views	AE	PSB	FT	43.3%	BoVF: 25.3%
Xie et al. [2015a]	descriptors	AEs	McGill	FT	78.2%	Lavoué [2012]: 55.7%
Liu et al. [2014]	descriptors	DBN	McGill	FT	61.81%	BoVF: 44.59%
Fang et al. [2015]	descriptors	Many-to-1 Encoder	McGill	PR Curves	-	[Tabia et al. 2014]
Han et al. [2016]	3D	MCRBMs	McGill	PR Curves	-	LFD, Spin Image, SI-HKS, **3D ShapeNets**
Bu et al. [2014]	descriptors	DBN	SHREC'11	FT	94.1%	Lian et al. [2011]: 40.6% Smeets et al. [2009]: 96.2%
Wang et al. [2015]	2D views	Siamese CNNs	SHREC'13	mAP	46.9%	[Li et al. 2014a]: 11.6%
Bai et al. [2016]	2D views	CNN	ModelNet40	mAP	81.94%	LFD: 40.91%
Su et al. [2015]	2D views	MVCNN	ModelNet40	mAP	79.5%	SPH: 33.26%
Wu et al. [2015]	3D	CDBN	ModelNet40	mAP	49.23%	
Feng et al. [2016]	RGB-D	CAEs Ensemble	ModelNet/ NYUv2	mAP	24%	HOG+l_2: 13% **Global AE**: 19%
Wang et al. [2016]	3D	CAE-ELM	ModelNet	PR Curves	-	LFD, SPH, **3D ShapeNets**

where: FT = First Tier (higher is better), mAP = mean Average Precision (higher is better), PR Curves = Precision-Recall Curves. *Note*: Methods in the last column highlighted with bold are "deep."

used and finally, the performance of the proposed approach and of other state-of-the-art methods it was compared to. Regarding 3D shape retrieval, AEs are a typical choice. As can be seen in Table I, Leng et al. [2015a] reported a First Tier (FT) of 67.2% on the Princeton Shape Benchmark (PSB) dataset surpassing by a large margin other deep and shallow methods. FT measures the ratio of models in the query's class that also appear within the top K matches (where K is the size of the query's class). On the popular ModelNet40 dataset, Bai et al. [2016] recently achieved a mean Average Precision (mAP) of 81.94% improving by almost 2.5% the previous highest performance of 79.5% by Su et al. [2015]. DBNs and CNNs have been used extensively in 3D object classification and recognition. Toward 3D instance recognition, Zaki et al. [2016] reported 97.2% accuracy on the popular RGB-D Object Dataset [Lai et al. 2011] outperforming other deep methods (e.g., Schwarz et al. [2015] with 94.1% accuracy) and state-of-the-art shallow methods, such as Bo et al. [2013] with 92.8% accuracy. In the context of class recognition, Eitel et al. [2015] reported 91.3±1.4 accuracy achieving state-of-the-art performance on the RGB-D Object Dataset using a two-stream CNN. As shown in Table II, Zaki et al. [2016] later achieved a similar performance (91.1±1.4) on the same dataset exploiting both depth and point cloud data.

ModelNet is the largest dataset available for 3D object classification, recognition, and retrieval containing more than 120,000 CAD models from 662 categories. As a consequence, ModelNet's two subsets, that is, ModelNet10 and ModelNet40, are used in the experimental evaluation of the majority of the latest published works. The first result reported on ModelNet [Wu et al. 2015] was 77.32% classification accuracy using ModelNet40, while the handcrafted descriptors Spherical Harmonic (SPH) [Kazhdan et al. 2003] and LFD combined with an SVM classifier achieved 68.23% and 75.47%, respectively. The margin was even bigger for the retrieval task with 3D ShapeNets [Wu et al. 2015] reaching 49.23% mAP on the same dataset followed by LFD with

Table II. Deep Learning Architectures for 3D Object Classification/Recognition

			Experimental		
Method	**Input**	**Deep Model**	**Dataset**	**Accuracy**	**Other Methods (shallow)**
Brock et al. [2016]	3D	CNN	ModelNet40	95.54%	LFD: 75.47% SPH: 68.23%
Qi et al. [2016]	2D views	MVCNN		91.4%	
Hegde and Zadeh [2016]	2D views + 3D	CNN		90.8%	
Johns et al. [2016]	2D views	CNN	ModelNet40	90.7%	
Su et al. [2015]	2D views	MVCNN		90.1%	
Wang et al. [2016]	3D	CAE-ELM		84.35%	
Maturana and Scherer [2015]	3D	3D CNN	ModelNet40	83%	
Xie et al. [2015b]	2D views	MVD-ELM		81.39%	
Wu et al. [2015]	3D	CDBN		77.32%	
			ModelNet10	83.5%	LFD: 79.87% SPH: 79.79%
Sedaghat et al. [2016]	3D	3D CNN		93.8%	
Leng et al. [2014]	2D views	DBN	PSB	78.3%	EW-GF: 71.4%, CMVD-GL:66.1%
Liu et al. [2014]	descriptors	DBN	SHREC'07 watertight	93%	BoVF: 83% PCA: 78.5% LDA: 71.5%
Bu et al. [2014]	descriptors	DBN	SHREC'07 watertight	85%	
Bu et al. [2015]	descriptors	DBN	McGill	87.5%	-
Zaki et al. [2016]	RGB-D + 3D	CNN	RGB-D object dataset	97.2% (instance)	[Bo et al. 2013]: 92.8% [Lai et al. 2011]: 73.9%
Schwarz et al. [2015]	RGB-D	CNN		94.1% (instance)	
Zaki et al. [2016]	RGB-D + 3D	CNN	RGB-D object dataset	91.1±1.4 (category)	[Bo et al. 2013]: 87.5±2.9 [Lai et al. 2011]: 81.9±2.8
Eitel et al. [2015]	RGB-D	CNN		91.3±1.4	
Schwarz et al. [2015]	RGB-D	CNN		89.4±1.3 (category)	
Socher et al. [2012]	RGB-D	CNNs & RNNs		86.8±3.3	

40.91% and SPH with only 33.26%. Later, view-based methods seemed to perform better at least in the scenario of 3D object classification reaching an accuracy of over 90% on ModelNet40 (Su et al. [2015] and Qi et al. [2016]) and over 92% on ModelNet10 (Johns et al. [2016]). Attempts to fuse the benefits of 2D views and 3D data were also made leading to satisfying results (e.g., Hegde and Zadeh [2016]), however, a recently published work (Brock et al. [2016]) demonstrated the power of volumetric models achieving state-of-the-art classification accuracy on both ModelNet40 (i.e., 95.54%) and ModelNet10 (i.e., 97.14%). Opting to visualize the advancements in 3D classification, the performance evolution (in terms of accuracy) through time of various methods on the two ModelNet subsets is shown in Figure 9. The diagram shows the significant boost that DNNs provide in comparison to shallow methods. At the same time, the rapid progress being made in the field of DNNs is highlighted, as the classification accuracy on ModelNet40 and ModelNet10 improved by almost 20% and 15%, respectively, within the last year and by 30% and 20% compared to the shallow representation methods. The significant gain introduced by DNNs is also apparent in Table III where deep networks are compared with shallow, SVM-based methods in the context of hyperspectral data classification resulting in higher performance.

Table III. Deep Learning Architectures Addressing Hyperspectral Data Classification

		Experimental		
Method	**Deep Model**	**Dataset**	**Accuracy (%)**	**Other Methods**
Chen et al. [2014]	SAEs	Kennedy Space Center	97.9	RBF-SVM: 97.69 EMP-RBF-SVM: 97.04
Chen et al. [2016]	CNN	Indian Pines	99.23±0.19	3D-RBF-SVM: 92.14±1.44 3D-EMP-RBF-SVM: 95.07±0.88
Makantasis et al. [2015]	CNN	Indian Pines	98.88	R-PCA RBF-SVM: 82.71 RBF-SVM: 82.79 Linear SVM: 79.56%
Chen et al. [2015b]	DBN	Indian Pines	95.45±0.18	RBF-SVM: 95.12±0.13 EMP: 94.71±0.25
Hu et al. [2015]	CNN	Indian Pines	90.16	RBF-SVM: 87.6 **LeNet-5 (deep)**: 88.27
Zhou et al. [2016]	SNN	Indian Pines	85.48±2.1	KELM: 50.67±2.46 SVM: 53.51±1.58
Zhao et al. [2016]	multiscale CNN	Pavia University	98.38	SVM: 88.33 EMP: 85 CK-SVM: 91.98
Yue et al. [2016]	SAEs and CNNs	Pavia University	96.31±1.14	RBF-SVM: 91.46±0.74 EMP-SVM: 89.27±1.5

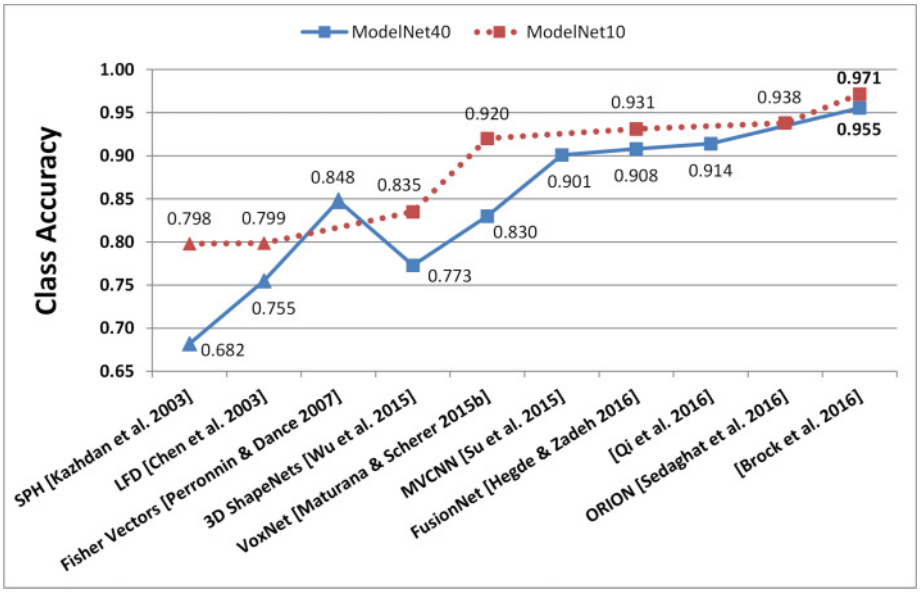


Fig. 9. Class Accuracy improvement on ModelNet40 and ModelNet10 datasets (triangle points indicate "nondeep" methods).

Overall, based on the findings of this survey, it can be concluded that DL can provide effective solutions for improving the performance of computer vision techniques in the 3D domain as well. In the electronic appendix, a short presentation of open-source 3D processing and DL libraries is available. Moreover, any software package provided by scientists in the context of the publications covered in this section are reported for potential use and experimentation along with a list of popular 3D datasets.

5. CONCLUSION AND DISCUSSION

The ongoing evolution of new capturing devices with more powerful capabilities poses new opportunities and at the same time new challenges to the 3D computer vision community. DL managed to revolutionize many 2D computer vision tasks reaching or even surpassing human-level performance and now it has started to be tested in the 3D domain as well. Even though 3D data offer a more accurate and discriminative representation of the captured scene, its intrinsic complex structure makes their deployment in deep architectures not trivial. In this work, we classified methods applying DL techniques on 3D data into five categories based on how they treated the input data before feeding them to the employed DNN. Experimental results indicate in general a slight advantage of methods exploiting multiple 2D views for representing the 3D scene in comparison to those taking advantage of the full 3D geometry. A recent work managed to achieve superior performance utilizing volumetric (i.e., 3D) models; however, a more complex architecture was proposed and notable data augmentation was required. In addition, despite the benefits obtained from GPUs rapid evolution, the computational cost of processing 3D data is a significant bottleneck that forces researchers to work on low-resolution 3D point clouds, a factor which has recently been shown to have an important impact on performance. Further emphasis should be placed on this observation in order to examine the extent of its influence on volumetric neural networks and explore potential solutions.

Speeding up neural networks training and model compression are currently attracting a lot of interest. Decreasing the parameters of a network can lead to faster training and at the same time increases the possibility of utilizing DL techniques in devices with limited resources. Another current research trend, not applied yet in 3D, is the use of spatial RNNs, especially LSTM-based models. LSTMs have several desired properties, for example, they can be fine-tuned end-to-end and they allow variable lengths in input and output. Based on neuroscience indications that recurrent synapses do exist in human brain, RNNs have recently been used in visual recognition, image/video description, and scene labeling often combined with CNNs. Finally, methods for visualizing the responses of DNNs have emerged recently. Since neural networks become deeper and more sophisticated, it is difficult to fully understand the performed operations. By visualizing how a network responds to a specific input, the opportunity to guide and improve its training process or its design arises. Research toward visualization is now limited but active, hence more advancements are expected in the near feature.

REFERENCES

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Man, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Vigas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. 2012. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 11 (2012), 2274–2282.

A. Agarwal, E. Akchurin, C. Basoglu, G. Chen, S. Cyphers, J. Droppo, A. Eversole, B. Guenter, M. Hillebrand, R. Hoens, X. Huang, Z. Huang, V. Ivanov, A. Kamenev, P. Kranen, O. Kuchaiev, W. Manousek, A. May, B. Mitra, O. Nano, G. Navarro, A. Orlov, M. Padmilac, H. Parthasarathi, B. Peng, A. Reznichenko, F. Seide, M. L. Seltzer, M. Slaney, A. Stolcke, Y. Wang, H. Wang, K. Yao, D. Yu, Y. Zhang, and G. Zweig. 2014. *An Introduction to Computational Networks and the Computational Network Toolkit*. Technical Report MSR-TR-2014-112. Microsoft Research.

A. K. Aijazi, P. Checchin, and L. Trassoudaine. 2013. Segmentation based classification of 3D urban point clouds: A super-voxel based approach with evaluation. *Remote Sensing* 5, 4 (2013), 1624–1650.

A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze. 2012a. A global hypotheses verification method for 3D object recognition. In *Proceedings of the 12th European Conference on Computer Vision*. 511–524.

A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze. 2012b. *Pattern Recognition: Joint 34th DAGM and 36th OAGM Symposium*. Chapter OUR-CVFH – Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation, 113–122.

A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski. 2011. CAD-model recognition and 6DOF pose estimation using 3D cues. In *IEEE ICCV Workshops*. 585–592.

L. A. Alexandre. 2012. 3D descriptors for object and category recognition: A comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ IROS*.

L. A. Alexandre. 2014. 3D Object recognition using convolutional neural networks with transfer learning between input channels. In *13th International Conference on Intelligent Autonomous Systems*, Vol. 301.

S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah. 2015. Comparative study of Caffe, Neon, Theano, and Torch for deep learning. *CoRR* abs/1511.06435 (2015).

S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Jan Latecki. 2016. GIFT: A real-time and scalable 3D shape search engine. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

P. Baldi and P. J. Sadowski. 2013. Understanding dropout. In *Advances in Neural Information Processing Systems 26*. 2814–2822.

F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. 2012. Theano: New features and speed improvements. *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*.

S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick. 2015. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *CoRR* abs/1512.04143 (2015).

J. A. Benediktsson, J. A. Palmason, and J. Sveinsson. 2005. Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE TGRS* 43, 3 (2005), 480–491.

Y. Bengio. 2012. *Neural Networks: Tricks of the Trade: Second Edition*. Chapter: Practical Recommendations for Gradient-Based Training of Deep Architectures, 437–478.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*. 153–160.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. 2011. Algorithms for hyper-parameter optimization. In *25th Annual Conference on Neural Information Processing Systems (NIPS 2011)*, Vol. 24.

J. Bergstra and Y. Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research* 13 (2012), 281–305.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: A CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation.

P. J. Besl and N. D. McKay. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 2 (1992), 239–256.

J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot. 2013. Hyperspectral remote sensing data analysis and future challenges. *IEEE Geoscience and Remote Sensing Magazine* 1, 2 (2013), 6–36.

L. Bo, X. Ren, and D. Fox. 2013. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics: The 13th International Symposium on Experimental Robotics*. 387–402.

D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter. 2011. The 3D Hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research* 2, 2 (2011), 1–13.

F. Bosche, Y. Turkan, C. Haas, and R. Haas. 2010. Fusing 4D modeling and laser scanning for automated construction progress control. *26th ARCOM Annual Conference and Annual General Meeting* (2010).

Y.-L. Boureau, J. Ponce, and Y. LeCun. 2010. A theoretical analysis of feature pooling in vision algorithms. In *Proceedings of the International Conference on Machine learning (ICML'10)*.

A. Brock, Th. Lim, J. M. Ritchie, and N. Weston. 2016. Generative and discriminative voxel modeling with convolutional neural networks. *CoRR* abs/1608.04236 (2016).

M. M. Bronstein and I. Kokkinos. 2010. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*. 1704–1711.

S. Bu, P. Han, Z. Liu, J. Han, and H. Lin. 2015. Local deep feature learning framework for 3D shape. *Computers & Graphics* 46 (2015), 117–129. Shape Modeling International 2014.

S. Bu, Z. Liu, J. Han, J. Wu, and R. Ji. 2014. Learning high-level feature by deep belief networks for 3-D model retrieval and recognition. *IEEE Transactions on Multimedia* 16, 8 (2014), 2154–2167.

B. Bustos, D. Keim, D. Saupe, and T. Schreck. 2007. Content-based 3D object retrieval. *IEEE Computer Graphics and Applications* 27, 4 (2007), 22–27.

W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. 2015. Scene labeling with LSTM recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 3547–3555.

Z. Cai, J. Han, L. Liu, and L. Shao. 2016. RGB-D datasets using microsoft kinect or similar sensors: A survey. *Multimedia Tools and Applications* (2016), 1–43.

N. Charbonneau, J. Burgess, and L. Robichaud. 2015. Using 4D modelling in a university-museum research partnership. In *2015 Digital Heritage*, Vol. 2. 603–610.

K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*.

D.-Y. Chen, X. P. Tian, Y.-T. Shen, and M. Ouhyoung. 2003. On visual similarity based 3D model retrieval. *Computer Graphics Forum (EUROGRAPHICS'03)* 22, 3 (2003), 223–232.

H. Chen and B. Bhanu. 2007. 3D free-form object recognition in range images using local surface patches. *Pattern Recognition Letters* 28, 10 (2007), 1252–1262.

W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. 2015a. Compressing neural networks with the hashing trick. *CoRR* abs/1504.04788 (2015).

Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi. 2016. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE TGRS* 54, 10 (2016), 6232–6251.

Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu. 2014. Deep learning-based classification of hyperspectral data. *IEEE J-STARS* 7, 6 (2014), 2094–2107.

Y. Chen, X. Zhao, and X. Jia. 2015b. Spectral-spatial classification of hyperspectral data based on deep belief network. *IEEE J-STARS* 8, 6 (2015), 2381–2392.

R. Collobert, K. Kavukcuoglu, and C. Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

R. Collobert, K. Kavukcuoglu, and C. Farabet. 2012. *Neural Networks: Tricks of the Trade: Second Edition*. Chapter: Implementing Neural Networks Efficiently, 537–557.

C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297.

C. Couprie, C. Farabet, L. Najman, and Y. Lecun. 2013. Indoor semantic segmentation using depth information. *CoRR* abs/1301.3572 (2013).

P. Daras and A. Axenopoulos. 2010. A 3D shape retrieval framework supporting multimodal queries. *International Journal of Computer Vision* 89, 2 (2010), 229–247.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *IEEE Computer Vision and Pattern Recognition (CVPR'09)*.

L. Deng. 2014. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing* 3 (2014), e5.

M. Denil, B. Shakibi, L. Dinh, M. A. Ranzato, and N. de Freitas. 2013. Predicting parameters in deep learning. *CoRR* abs/1306.0543 (2013).

E. Denton, E. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. *CoRR* abs/1404.0736 (2014).

B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. 2011. On the segmentation of 3D LIDAR point clouds. In *IEEE ICRA*. 2798–2805.

A. Doulamis, M. Ioannides, N. Doulamis, A. Hadjiprocopis, D. Fritsch, O. Balet, M. Julien, E. Protopapadakis, and others. 2013. 4D reconstruction of the past. *Proceedings of SPIE* 8795 (2013), 87950J-1–87950J-11.

A. Doulamis, S. Soile, N. Doulamis, C. Chrisouli, N. Grammalidis, K. Dimitropoulos, C. Manesis, C. Potsiou, and C. Ioannidis. 2015. Selective 4D modelling framework for spatial-temporal land information management system. *Proceedings of SPIE 9535, 3rd RSCy* (2015).

N. Doulamis and A. Doulamis. 2012. Fast and adaptive deep fusion learning for detecting visual objects. In *Proceedings of ECCV 2012. Workshops and Demonstrations*. 345–354.

A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. 2015. Multimodal deep learning for robust RGB-D object recognition. In *IEEE/RSJ International Conference on IROS*.

Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong. 2015. 3D deep shape descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 2319–2328.

M. Fauvel, J. Chanussot, and J. A. Benediktsson. 2012. A spatial–spectral kernel-based approach for the classification of remote-sensing images. *Pattern Recognition* 45, 1 (2012), 381–392.

J. Feng, Y. Wang, and S.-F. Chang. 2016. 3D shape retrieval using single depth image from low-cost sensors. In *IEEE Winter Conference on Applications of Computer Vision (WACV'16)*.

S. Filipe and L. A. Alexandre. 2014. A comparative evaluation of 3D keypoint detectors in a RGB-D object dataset. In *9th International Conference on Computer Vision Theory and Applications*. 476–483.

S. Filipe, L. Itti, and L. A. Alexandre. 2015. BIK-BUS: Biologically motivated 3D keypoint based on bottom-up saliency. *IEEE Transactions on Image Processing* 24, 1 (2015), 163–175.

A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. 2004. Recognizing objects in range data using regional point descriptors. In *ECCV 2004*. Lecture Notes in Computer Science, Vol. 3023. 224–237.

Y. Gao and Q. Dai. 2014. View-based 3D object retrieval: Challenges and approaches. *IEEE MultiMedia* 21, 3 (2014), 52–57.

Y. Gao, M. Wang, D. Tao, R. Ji, and Q. Dai. 2012. 3-D object retrieval and recognition with hypergraph analysis. *IEEE Transactions on Image Processing* 21, 9 (2012), 4290–4303.

Y. Gao, M. Wang, Z. J. Zha, Q. Tian, Q. Dai, and N. Zhang. 2011. Less is more: Efficient 3-D object retrieval with query view selection. *IEEE Transactions on Multimedia* 13, 5 (2011), 1007–1018.

D. Giorgi, S. Biasotti, and L. Paraboschi. 2007. Shape retrieval contest 2007: Watertight models track. *SHREC Competition* 8 (2007).

A. Godil, H. Dutagaci, C. Akgul, A. Axenopoulos, B. Bustos, M. Chaouch, P. Daras, and others. 2009. SHREC'09 track: Generic shape retrieval. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*. 61–68.

Y. Gong, L. Liu, M. Yang, and L. D. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *CoRR* abs/1412.6115 (2014).

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. 2013. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*. 1319–1327.

A. Graves, A. Mohamed, and G. E. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'13)*. 6645–6649.

K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. 2015. DRAW: A recurrent neural network for image generation. *CoRR* abs/1502.04623 (2015).

Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan. 2014. 3D object recognition in cluttered scenes with local surface features: A survey. *IEEE TPAMI* 36, 11 (2014), 2270–2287.

Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. Kwok. 2016a. A comprehensive performance evaluation of 3D local feature descriptors. *IJCV* 116, 1 (2016), 66–89.

Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. 2016b. Deep learning for visual understanding: A review. *Neurocomputing* 187 (2016), 27–48. Recent Developments on Deep Big Vision.

Y. Guo, F. A. Sohel, M. Bennamoun, M. Lu, and J. Wan. 2013. Rotational projection statistics for 3D local surface description and object recognition. *CoRR* abs/1304.3192 (2013).

Y. Guo, F. A. Sohel, M. Bennamoun, J. Wan, and M. Lu. 2015. A novel local surface feature for 3D object recognition under clutter and occlusion. *Information Sciences* 293 (2015), 196–213.

Y. Guo, J. Zhang, M. Lu, J. Wan, and Y. Ma. 2014. Benchmark datasets for 3D computer vision. In *9th IEEE Conference on Industrial Electronics and Applications (ICIEA'14)*. 1846–1851.

S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. 2014. Learning rich features from RGB-D images for object detection and segmentation. In *Proceedings of the 13th European Conference on Computer Vision*.

Z. Han, Z. Liu, J. Han, C. M. Vong, S. Bu, and C. L. P. Chen. 2016. Mesh convolutional restricted Boltzmann machines for unsupervised learning of features with structure preservation on 3-D meshes. *IEEE Transactions on Neural Networks and Learning Systems* PP, 99 (2016), 1–14.

K. He, X. Zhang, S. Ren, and J. Sun. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR* abs/1406.4729 (2014).

K. He, X. Zhang, S. Ren, and J. Sun. 2015a. Deep residual learning for image recognition. *CoRR* abs/1512.03385 (2015).

K. He, X. Zhang, S. Ren, and J. Sun. 2015b. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR* abs/1502.01852 (2015).

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Identity mappings in deep residual networks. *CoRR* abs/1603.05027 (2016).

V. Hegde and R. Zadeh. 2016. FusionNet: 3D object classification using multiple data representations. *CoRR* abs/1607.05695 (2016).

M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. 2001. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of the 28th SIGGRAPH*. 203–212.

G. E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14, 8 (2002), 1771–1800.

G. E. Hinton, P. Dayan, B. Frey, and R. M. Neal. 1995. The wake-sleep algorithm for self-organizing neural networks. *Science* 268, 5124 (1995), 1158–1161.

G. E. Hinton, S. Osindero, and Y.-W. Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18, 7 (2006), 1527–1554.

G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580 (2012).

G. E. Hinton, O. Vinyals, and J. Dean. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531 (2015).

S. Hochreiter. 1991. *Untersuchungen Zu Dynamischen Neuronalen Netzen*. Diploma thesis. Technical University Munich, Institute of Computer Science.

S. Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *IJUFKS* 6, 2 (1998), 107–116.

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

W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li. 2015. Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors* 2015, Article 258619 (2015).

G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. 2016. Deep networks with stochastic depth. *CoRR* abs/1603.09382 (2016).

G. B. Huang, H. Zhou, X. Ding, and R. Zhang. 2012. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42, 2 (2012), 513–529.

G. B. Huang, Q.-Y. Zhu, and C.-K. Siew. 2006. Extreme learning machine: Theory and applications. *Neurocomputing* 70, 13 (2006), 489–501.

M. Ioannides, A. Hadjiprocopis, N. Doulamis, A. Doulamis, E. Protopapadakis, K. Makantasis, and others. 2013. Online 4D reconstruction using multi-images available under open access. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences 1* (2013), 169–174.

S. Ioffe and C. Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR* abs/1502.03167 (2015).

M. Jaderberg, A. Vedaldi, and A. Zisserman. 2014. Speeding up convolutional neural networks with low rank expansions. *CoRR* abs/1405.3866 (2014).

K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. 2009. What is the best multi-stage architecture for object recognition? In *12th IEEE International Conference on Computer Vision*. 2146–2153.

S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani. 2006. Developing an engineering shape benchmark for CAD models. *Computer-Aided Design* 38, 9 (2006), 939–953.

H. Jegou, M. Douze, and C. Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093* (2014).

E. Johns, S. Leutenegger, and A. J. Davison. 2016. Pairwise decomposition of image sequences for active multi-view recognition. In *Proceedings of the IEEE Conference on CVPR*. 3183–3822.

A. E. Johnson and M. Hebert. 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 5 (1999), 433–449.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. 2014. A convolutional neural network for modelling sentences. *CoRR* abs/1404.2188 (2014).

L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong. 2013. Representational learning with extreme learning machine for big data. *IEEE Intelligent Systems* 28, 6 (2013), 31–34.

M. Kazhdan, Th. Funkhouser, and S. Rusinkiewicz. 2003. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*.

J. M. Khatib, N. Chileshe, and S. Sloan. 2007. Antecedents and benefits of 3D and 4D modelling for construction planners. *Journal of Engineering, Design and Technology* 5, 2 (2007), 159–172.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 1097–1105.

A. Krogh and J. A. Hertz. 1992. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, Vol. 4. 950–957.

G. Kyriakaki, A. Doulamis, N. Doulamis, M. Ioannides, K. Makantasis, E. Protopapadakis, A. Hadjiprocopis, K. Wenzel, and others. 2014. 4D reconstruction of tangible cultural heritage objects from web-retrieved images. *International Journal of Heritage in the Digital Era* 3, 2 (2014), 431–451.

L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. 2009. Associative hierarchical CRFs for object class image segmentation. *Proceedings of the IEEE* 12*th International Conference on Computer Vision* (2009).

K. Lai, L. Bo, X. Ren, and D. Fox. 2011. A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE International Conference on on Robotics and Automation*.

G. Lavoué. 2012. Combination of bag-of-words descriptors for robust partial shape retrieval. *The Visual Computer* 28, 9 (2012), 931–942.

V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky. 2014. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. *CoRR* abs/1412.6553 (2014).

Y. LeCun, Y. Bengio, and G. E. Hinton. 2015. Deep learning. *Nature* 521 (2015), 436–444.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of IEEE* 86, 11 (1998), 2278–2324.

Y. LeCun, K. Kavukcuoglu, and C. Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10)*. 253–256.

H. Lee, E. Chaitanya, and A. Y. Ng. 2008. Sparse deep belief net model for visual area V2. In *Advances in Neural Information Processing Systems 20*. 873–880.

B. Leng, S. Guo, X. Zhang, and Z. Xiong. 2015. 3D object retrieval with stacked local convolutional autoencoder. *Signal Processing* 112, C (2015), 119–128.

B. Leng, Y. Liu, K. Yu, X. Zhang, and Z. Xiong. 2016. 3D object understanding with 3D convolutional neural networks. *Information Sciences* 336, C (Oct. 2016), 188–201.

B. Leng, X. Zhang, M. Yao, and Z. Xiong. 2014. *MultiMedia Modeling: 20th Anniversary International Conference, Part II*. Chapter: 3D Object Classification Using Deep Belief Networks, 128–139.

B. Li, Y. Lu, A. Godil, T. Schreck, B. Bustos, A. Ferreira, and others. 2014a. A comparison of methods for sketch-based 3D shape retrieval. *Computer Vision and Image Understanding* 119 (2014), 57–80.

B. Li, Y. Lu, C. Li, and others. 2015. A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries. *Computer Vision and Image Understanding* 131 (2015).

B. Li, E. Zhou, B. Huang, J. Duan, Y. Wang, N. Xu, J. Zhang, and H. Yang. 2014b. Large scale recurrent neural network on GPU. In *International Joint Conference on Neural Networks (IJCNN'14)*. 4062–4069.

Z. Lian, A. Godil, B. Bustos, M. Daoudi, and others. 2011. SHREC'11 track: Shape retrieval on non-rigid 3D watertight meshes. In *Proceedings of the 4th Eurographics Conference on 3D Object Retrieval*. 79–88.

M. Lin, Q. Chen, and S. Yan. 2013. Network in network. *CoRR* abs/1312.4400 (2013).

Q. Liu. 2012. A survey of recent view-based 3D model retrieval methods. *CoRR* abs/1208.3670 (2012).

Z. Liu, S. Chen, S. Bu, and K. Li. 2014. High-level semantic feature for 3D shape based on deep belief networks. In *IEEE International Conference on Multimedia and Expo (ICME'14)*. 1–6.

D. G. Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV'99)*, Vol. 2. 1150–1157.

A. Maas, A. Hannun, and A. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the ACL*. 142–150.

A. Mademlis, P. Daras, D. Tzovaras, and M. G. Strintzis. 2009. 3D object retrieval using the 3D shape impact descriptor. *Pattern Recognition* 42, 11 (2009), 2447–2459.

K. Makantasis, A. Doulamis, N. Doulamis, and M. Ioannides. 2016. In the wild image retrieval and clustering for 3D cultural heritage landmarks reconstruction. *MTAP* 75, 7 (2016), 3593–3629.

K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis. 2015. Deep supervised learning for hyperspectral data classification through convolutional neural networks. In *IEEE IGARSS*. 4959–4962.

J. Martens and I. Sutskever. 2011. Learning recurrent neural networks with Hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 1033–1040.

H. P. Martínez and G. N. Yannakakis. 2014. Deep multimodal fusion: Combining discrete events and continuous signals. In *Proceedings of the 16th International Conference on Multimodal Interaction*. 34–41.

M. Mathieu, M. Henaff, and Y. LeCun. 2013. Fast training of convolutional networks through FFTs. *CoRR* abs/1312.5851 (2013).

D. Maturana and S. Scherer. 2015. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 922–928.

W. McCulloch and W. Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* 5, 4 (1943), 115–133.

A. Merentitis and C. Debes. 2015. Automatic fusion and classification using random forests and features extracted with deep learning. In *International Geoscience and Remote Sensing Symposium*. 2943–2946.

A. Mian, M. Bennamoun, and R. Owens. 2010. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *IJCV* 89, 2–3 (2010), 348–361.

K. Mikolajczyk and C. Schmid. 2005. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 10 (2005), 1615–1630.

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. 2005. A comparison of affine region detectors. *IJCV* 65, 1 (2005), 43–72.

M. Muja and D. G. Lowe. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP'09)*. 331–340.

V. Nair and G. E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 807–814.

A. Nguyen and B. Le. 2013. 3D point cloud segmentation: A survey. In *Proceedings of the 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. 225–230.

M. Niepert, M. Ahmed, and K. Kutzkov. 2016. Learning convolutional neural networks for graphs. *CoRR* abs/1605.05273 (2016).

W. Ouyang, P. Luo, X. Zeng, S. Qiu, Y. Tian, H. Li, S. Yang, and others. 2014. DeepID-Net: Multi-stage and deformable deep convolutional neural networks for object detection. *CoRR* abs/1409.3505 (2014).

J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. 2013. Voxel cloud connectivity segmentation—Supervoxels for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2027–2034.

R. Pascanu, C. Gülçehre, K. Cho, and Y. Bengio. 2013a. How to construct deep recurrent neural networks. *CoRR* abs/1312.6026 (2013).

R. Pascanu, T. Mikolov, and Y. Bengio. 2013b. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*. 1310–1318.

C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas. 2016. Volumetric and multi-view CNNs for object classification on 3D data. *arXiv preprint arXiv:1604.03265v2* (2016).

T. Rabbani, F. Van Den Heuvel, and G. Vosselmann. 2006. Segmentation of point clouds using smoothness constraint. *ISPRS Archives* 36, 5 (2006), 248–253.

M. Ranzato, Y. Boureau, and Y. LeCun. 2008. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems 20*. 1185–1192.

S. Ren, K. He, R. Girshick, and J. Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. 91–99.

S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. 2011. Contracting auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th ICML*. 833–840.

A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. 2014. FitNets: Hints for thin deep nets. *CoRR* abs/1412.6550 (2014).

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65, 6 (1958), 386–408.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323 (1986), 533–536.

R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. 2008. Aligning point cloud views using persistent feature histograms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3384–3391.

R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. 2010. Fast 3D recognition and pose using the viewpoint feature histogram. In *IEEE/RSJ International Conference on IROS*. 2155–2162.

R. B. Rusu and S. Cousins. 2011. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA'11)*. 1–4.

S. Salti, A. Petrelli, F. Tombari, and L. Di Stefano. 2012. On the affinity between 3D detectors and descriptors. In *Proceedings of the 2nd International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*. 424–431.

J. Sanchez-Riera, K.-L. Hua, Y.-S. Hsiao, T. Lim, S. C. Hidayati, and W.-H. Cheng. 2016. A comparative study of data fusion for RGB-D based visual recognition. *Pattern Recognition Letters* 73 (2016), 1–6.

D. Scherer, A. Muller, and S. Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *20th ICANN*. Vol. 6354. 92–101.

G. Schindler and F. Dellaert. 2012. 4D cities: Analyzing visualizing and interacting with historical urban photo collections. *Journal of Multimedia* (2012).

C. Schmid, R. Mohr, and C. Bauckhage. 2000. Evaluation of interest point detectors. *International Journal of Computer Vision* 37, 2 (2000), 151–172.

J. Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Computation* 4, 2 (1992), 234–242.

J. Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117.

R. Schnabel, R. Wahl, and R. Klein. 2007. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum* 26, 2 (2007), 214–226.

M. Schwarz, H. Schulz, and S. Behnke. 2015. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In *IEEE ICRA*. 1329–1335.

N. Sedaghat, M. Zolfaghari, and Th. Brox. 2016. Orientation-boosted voxel nets for 3D object recognition. *CoRR* abs/1604.03351 (2016).

P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. 2004. The Princeton shape benchmark. In *Shape Modeling International*.

K. Siddiqi, J. Zhang, D. Macrini, A. Shokoufandeh, S. Bouix, and S. Dickinson. 2008. Retrieving articulated 3-D models using medial surfaces. *Machine Vision and Applications* 19, 4 (2008), 261–275.

N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. 2012. Indoor segmentation and support inference from RGBD images. In *ECCV*.

M.-C. Sima and A. Nuchter. 2013. An extension of the Felzenszwalb-Huttenlocher segmentation to 3D point clouds. *5th ICMV: Computer Vision, Image Analysis and Processing* 8783 (2013).

D. Smeets, Th. Fabry, J. Hermans, D. Vandermeulen, and P. Suetens. 2009. Isometric deformation modelling for object recognition. In *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*. 757–765.

R. Socher, B. Huval, B. Bhat, C. D. Manning, and A. Y. Ng. 2012. Convolutional-recursive deep learning for 3D object classification. In *Advances in Neural Information Processing Systems 25*. 656–664.

S. Song and J. Xiao. 2014. Sliding shapes for 3D object detection in depth images. In *Proceedings of the 13th European Conference on Computer Vision (ECCV'14)*. 634–651.

S. Song and J. Xiao. 2016. Deep sliding shapes for amodal 3D object detection in RGB-D images. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition*.

H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. 2015. Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the International Conference on Computer Vision (ICCV'15)*.

I. Sutskever. 2012. *Training Recurrent Neural Networks*. Ph.D. dissertation. University of Toronto.

I. Sutskever, J. Martens, and G. E. Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 1017–1024.

C. Szegedy, S. Ioffe, and V. Vanhoucke. 2016. Inception-v4, inception-ResNet and the impact of residual connections on learning. *CoRR* abs/1602.07261 (2016).

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2014. Going deeper with convolutions. *CoRR* abs/1409.4842 (2014).

H. Tabia, H. Laga, D. Picard, and P.-H. Gosselin. 2014. Covariance descriptors for 3D shape matching and retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*. 4185–4192.

J. Tang, S. Miller, A. Singh, and P. Abbeel. 2012. A textured object recognition pipeline for color and depth image data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'12)*.

J. W. H. Tangelder and R. C. Veltkamp. 2007. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications* 39, 3 (2007), 441.

L. Theis and M. Bethge. 2015. Generative image modeling using spatial LSTMs. In *Advances in Neural Information Processing Systems 28*.

F. Tombari and L. Di Stefano. 2012. Hough voting for 3D object recognition under occlusion and clutter. *IPSJ Transactions on Computer Vision and Applications* 4 (2012), 20–29.

F. Tombari, S. Salti, and L. Di Stefano. 2010. Unique signatures of histograms for local surface description. In *Proceedings of the 11th European Conference on Computer Vision: Part III (ECCV'10)*. 356–369.

F. Tombari, S. Salti, and L. Di Stefano. 2013. Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision* 102, 1–3 (2013), 198–220.

J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. 2013. Selective search for object recognition. *International Journal of Computer Vision* (2013).

J. P. C. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P. H. S. Torr. 2013. Mesh based semantic modelling for indoor and outdoor scenes. In *IEEE CVPR*. 2067–2074.

V. Vanhoucke, A. Senior, and M. Z. Mao. 2011. Improving the speed of neural networks on CPUs. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*.

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11 (2010), 3371–3408.

F. Visin, K. Kastner, K. Cho, M. Matteucci, A. C. Courville, and Y. Bengio. 2015. ReNet: A recurrent neural network based alternative to convolutional networks. *CoRR* abs/1505.00393 (2015).

A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto. 2015. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing* 104 (2015), 88–100.

L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of the 30th ICML*, Vol. 28. 1058–1066.

F. Wang, L. Kang, and Y. Li. 2015. Sketch-based 3D shape retrieval using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

W. Wang, L. Chen, Z. Liu, K. Kühnlenz, and D. Burschka. 2013. Textured/textureless object recognition and pose estimation using RGB-D image. *Journal of Real-Time Image Processing* 10, 4 (2013), 667–682.

Y. Wang, Z. Xie, K. Xu, Y. Dou, and Y. Lei. 2016. An efficient and effective convolutional auto-encoder extreme learning machine network for 3D feature learning. *Neurocomputing* 174 (2016), 988–998.

D. Weikersdorfer, D. Gossow, and M. Beetz. 2012. Depth-adaptive superpixels. In *21st International Conference on Pattern Recognition (ICPR'12)*. 2087–2090.

P. J. Werbos. 1990. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE* 78, 10 (1990), 1550–1560.

W. Wohlkinger and M. Vincze. 2011. Ensemble of shape functions for 3D object classification. In *IEEE International Conference on Robotics and Biomimetics (ROBIO'11)*. 2987–2992.

H. Wu and X. Gu. 2015. Towards dropout training for convolutional neural networks. *Neural Networks* 71 (2015), 1–10.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 2015. 3D shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1912–1920.

J. Xie, Y. Fang, F. Zhu, and E. Wong. 2015a. DeepShape: Deep learned shape descriptor for 3D shape matching and retrieval. In *Proceedings of the IEEE Conference on CVPR*. 1275–1283.

Z. Xie, K. Xu, W. Shan, L. Liu, Y. Xiong, and H. Huang. 2015b. Projective feature learning for 3D shapes with multi-view depth images. *Computer Graphics Forum (Proceedings of Pacific Graphics 2015)* 34, 6 (2015).

B. Xu, N. Wang, T. Chen, and M. Li. 2015b. Empirical evaluation of rectified activations in convolutional network. *CoRR* abs/1505.00853 (2015).

Q. Xu, S. Jiang, W. Huang, F. Ye, and S. Xu. 2015a. Feature fusion based image retrieval using deep learning. *Journal of Information and Computational Science* 12, 6 (2015), 2361–2373.

Z. Yan, H. Zhang, Y. Jia, Th. Breuel, and Y. Yu. 2016. Combining the best of convolutional layers and recurrent layers: A hybrid network for semantic segmentation. *CoRR* abs/1603.04871 (2016).

J. Yue, S. Mao, and M. Li. 2016. A deep learning framework for hyperspectral image classification using spatial pyramid pooling. *Remote Sensing Letters* 7, 9 (2016), 875–884.

A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud. 2009. Surface feature detection and description with applications to mesh matching. In *IEEE Conference on CVPR*. 373–380.

H. F. M. Zaki, F. Shafait, and A. Mian. 2016. Convolutional hypercube pyramid for accurate RGB-D object category and instance recognition. In *IEEE ICRA*. 1685–1692.

D. Zarpalas, P. Daras, A. Axenopoulos, D. Tzovaras, and M. G. Strintzis. 2006. 3D model search and retrieval using the spherical trace transform. *EURASIP Journal on Advances in Signal Processing* 2007 (2006).

M. D. Zeiler and R. Fergus. 2013. Stochastic pooling for regularization of deep convolutional neural networks. *CoRR* abs/1301.3557 (2013).

A. Zelener. 2015. Survey of object classification in 3D range scans. (2015).

L. Zhang, L. Zhang, and B. Du. 2016a. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine* 4, 2 (2016), 22–40.

X. Zhang, H. Zhang, Y. Zhang, Y. Yang, M. Wang, H. Luan, J. Li, and T. S. Chua. 2016b. Deep fusion of multiple semantic cues for complex event recognition. *IEEE TIP* 25, 3 (2016), 1033–1046.

X. Zhang, J. Zou, X. Ming, K. He, and J. Sun. 2014. Efficient and accurate approximations of nonlinear convolutional networks. *CoRR* abs/1411.4229 (2014).

W. Zhao and S. Du. 2016. Learning multiscale and deep representations for classifying remotely sensed imagery. *ISPRS Journal of Photogrammetry and Remote Sensing* 113 (2016), 155–165.

Y. Zhong. 2009. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In *12th IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 689–696.

Y. Zhou and Y. Wei. 2016. Learning hierarchical spectral-spatial features for hyperspectral image classification. *IEEE Transactions on Cybernetics* 46, 7 (2016), 1667–1678.

Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai. 2014. Deep learning representation using autoencoder for 3D shape retrieval. *CoRR* abs/1409.7164 (2014).

Received June 2016; revised December 2016; accepted January 2017