# 3D computer vision based on machine learning with deep neural networks: A review

**Kailas Vodrahalli**
**Achintya K. Bhowmik** (SID Fellow)

***Abstract —*** Recent advances in the field of computer vision can be attributed to the emergence of deep learning techniques, in particular convolutional neural networks. Neural networks, partially inspired by the brain's visual cortex, enable a computer to "learn" the most important features of the images it is shown in relation to a specific, specified task. Given sufficient data and time, (deep) convolutional neural networks offer more easily designed, more generalizable, and significantly more accurate end-to-end systems than is possible with previously employed computer vision techniques. This review paper seeks to provide an overview of deep learning in the field of computer vision with an emphasis on recent progress in tasks involving 3D visual data. Through a backdrop of the mammalian visual processing system, we hope to also provide inspiration for future advances in automated visual processing.

***Keywords*** *— computer vision, artificial intelligence, machine learning, deep neural networks.*
DOI # 10.1002/jsid.617

## 1 Introduction

In the past decade, rapid advances in machine learning have helped us create self-driving cars[1] and autonomous drones,[2] enabled accurate recognition of speech,[3] translation of text,[4] and natural human-computer interfaces,[5] as well as shown promise in a wide variety of medical applications including drug research[6] and disease diagnosis.[7] Machine learning, in short, enables one to "learn" the most relevant statistics in relation to a predefined problem through analysis of sample ("training") inputs. As opposed to the alternative of hand-engineering a best guess of these statistics, machine learning gives a generalized framework that often performs better. Machine learning techniques can be effectively applied to a wide range of problems that require prediction of unobserved data with a highly intricate relation to the observed data. Given the incredible amount, complexity, and importance of data we accumulate in the modern era, it is easy to see why machine learning has become so popular.

Recent advances in automated visual and language understanding, among other areas, are by and large due to (deep) neural networks. Neural networks are a specific type of machine learning framework that is modeled roughly after the neurons in the brain. Input data is transformed by various operations through a series of layers – an input layer, at least one "hidden" layer not seen by the interfaces of the network, and an output layer. The output layer contains the useful information uncovered by the network. Deep neural networks are neural networks with many hidden layers. A hidden layer can be described as performing a fixed operation on its input, with variable parameter values learned through training. In practice, the hidden layers often perform matrix multiplications along with a variety of nonlinear operations; the matrix is filled with real-valued numbers learned through training. In order to perform well, a deep neural network will often have $>10^7$ tunable parameters. To train these parameters requires significant amounts of training data – often $>10^6$ individual training inputs – and much computational power. In addition, neural networks are finicky to train. A number of tricks to solve problems such as overfitting of training data (which prevent the network from generalizing when given test-time inputs) and prohibitively long training times needed to be developed to make neural networks practical. With the emergence of GPU hardware,[8] the rise in the amount of available training data, and breakthrough techniques enabling efficient and effective training,[9–13] neural networks have become a necessity in many areas of computer understanding and prediction systems. In this review, we will focus most of our attention to convolutional neural networks (CNNs), a specific type of deep neural network, due to their critical importance to computer vision.

Computer vision refers to a variety of tasks related to giving a computer the ability to "understand" what one or more cameras "see." Typical problems include detection of same points between two images of a single scene, image and scene classification (is this a cat?; is this in the bedroom?), object detection (roughly where is an object?) and segmentation (which pixels, exactly, belong to an object?), and face recognition among others. Prior to CNNs, the standard computer vision pipeline involved the following steps: (1) manually engineer a feature detector or specific algorithm (for example, to detect high-contrast edges for image segmentation) (2) potentially apply machine learning techniques (for example, for classifying the extracted features from the feature detector). A neural

Received 10/08/17; accepted 10/26/17.
Kailas Vodrahalli is with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, California, USA.
Achintya K. Bhowmik is with Starkey Hearing Technologies, Berkeley, California, USA; e-mail: achintya.k.bhowmik@gmail.com.

network allows for a full end-to-end solution that learns an optimal feature detector or algorithm for the given task, absolving the need for much of the manual engineering required in step 1. The result is a more general, more easily engineered, and often much more accurate system.

While much work has been carried out to enable understanding of 2D red-green-blue (RGB) images through CNNs and other means, 3D computer vision is a relatively new area. 3D computer vision involves understanding a 3D scene where we observe a color image with additional information on the depth of a scene. Most tasks of 3D computer vision have analogies in 2D computer vision.

In this paper, we give an overview of the current state of 3D computer vision – what the major accomplishments are and what are still major challenges. We also hope to provide a broader view to the field overall by incorporating perspectives from the field of neuroscience and psychology to understand our own biological visual system. We believe that this perspective is necessary to continue to make progress in automated (visual) understanding. Given the billions of years spent on developing the visual systems we are born with and their clear success, we can safely assume that there is a lot to learn from biological vision.

### 1.1 Related work

There are relatively few published works summarizing the state of affairs for 3D computer vision. The most relevant is a recent survey covering the advances of 3D computer vision due to deep learning, and the details of the techniques and network architectures used to make these advances.[14] We seek to provide a higher-level view of the field with a purpose of providing broader understanding and inspiration for future work in computer vision and related fields. We give brief overviews of topics like training deep neural networks, which we trust the reader can find abundant resources on, and we focus more on what the neural networks can do as well as specific details relevant to 3D computer vision. There have also been several important advances in the field since the last cited work of the review, which we direct our attention to in this review.

## 2 Neural networks and the computer vision problem

### 2.1 Biological basis for neurons

In the biological setting, a neuron is the basic unit of the brain. A neuron receives inputs from many surrounding neurons; if the stimulus reaches a threshold value, it then transmits an action potential to stimulate the neurons it connects to. There are a few key abstractions we can make to simplify the complex biology involved in a neuron. First, a single neuron receives inputs from many other neurons and transmits its action potential to many other neurons; there is a high connectivity between neurons. Second, the action potential is triggered at a threshold value; on a given input, only a portion of the neurons will activate indicating that the arrangement and relation of neurons is able to capture information about the structure of the input. Lastly, neurons are highly nonlinear; the transmitting of the signal from one neuron to the next is governed by the complex biochemistry at the synapses of the neurons. The neurons used in artificial neural networks, although much simplified from a true model of a neuron, are based in the aforementioned three abstractions.

### 2.2 Artificial neural networks

A neural network consists of a sequence of layers, each of which performs some operation on the output of the previous layer to produce a new output. The operation performed in each layer is fixed, while the parameters of the operation are variables that get tuned during training. The first layer is the input data in a vectorized form, the last layer is whatever prediction the network has been trained to output, and the middle layers perform the computations transforming the input to the output. These middle layers are called "hidden" layers, as they are not seen by the interfaces to the network. Each unit that produces a single output within a layer is called a neuron. The high connectivity of neurons is replicated by the operations within a layer. The input to a neuron is typically a combination of the outputs from many neurons in the preceding layer; the output of a neuron influences many neurons in the following layer.

A deep neural network is a neural network with many hidden layers. The intuitive benefit of these extra layers is twofold. First, there are now more parameters that can be trained to better learn and model the structure of the input data. Second, a deeper network allows for more hierarchical learning. The first layer can learn something very low level about the data, while later layers learn more abstract concepts about the data, culminating in the final output. For image data, the first layer might act as an edge detector, while subsequent layers detect compositions of edges for object detection.

The computations performed in the layers are somewhat arbitrary; however, most networks use only a few types of layers that work well in practice. A typical layer consists of a matrix multiplication, where the values in the matrix have been learned, followed by some a nonlinear activation function. The layer with the matrix multiplication is called a "fully-connected layer" as it stimulates a connection between each neuron in the given layer with each neuron in the following layer. While the main computation occurs in the matrix multiplication, the nonlinearity is necessary in order to connect two layers together. Because matrix multiplication is a linear operation, the multiple layers could be condensed into a single layer if there was no nonlinearity separating them. A popular example for the nonlinear function is the ReLU, which computes max(0, x) for each input x. The activation function is analogous to the action potential of a neuron; however, an activation function's main purpose is to introduce nonlinearity into the network. An additional operation commonly used in

computer vision is pooling, which reduces the size of the output by condensing fixed-size groups of input values into a single output value. A popular pooling function is max pooling, which reduces each group to the maximum value of the group.

We note here that neural networks are able to approximate any continuous function.[15] Although this ability is not unique to neural networks and is not the main reason they are so commonly used, it does provide some theoretical basis for their applicability to a wide variety of problem spaces. The reason they are so popular is their ability to generalize beyond training data in practical settings. For example, consider the problem of image recognition. For a 100 × 100 RGB image, there are well over $10^{11}$ possible images. The amount of training data may be on the order of $10^6$, substantially less than the possible images; yet, the neural network is able to capture the information that is important to generalizing in the natural world. It is for this reason that they are so useful.

Some additional insight into why deep networks in particular are useful is that they learn abstractions hierarchically. There is also some theoretical work that shows deeper networks can be exponentially better at approximating a function than shallower networks.[16]

## 2.3 Training a neural network

Training a neural network involves optimizing the values of the variable parameters in the network's hidden layers in order to minimize a predefined error (or cost) function. The error function is designed by the network architect to push a network to learn a specific task. In order to illustrate this process, we take an example from the problem of object classification. Let us assume that there are two classes of object we want our network to recognize: "bird" or "not bird." We will input an image to the network, and the output will be two probabilities, one giving the probability the image is of a bird, the other being the probability the image is not of a bird. Now there is only one correct answer – one probability should be 1, the other should be 0. In this sort of problem, we typically use annotated training data that has the correct answer associated with it. This setting is called supervised learning, as we can tell the network the correct answer and supervise the learning process. In order for the network to learn, we need to give it some feedback on the degree of its correctness. Note that the network's output will not typically be a black-and-white output but rather a sequence of probabilities, from which we take the answer with the highest probability.

Now to improve our network (reduce its error), we will modify its parameters. A common approach uses the backpropagation method.[17] We input the network's output along with the correct, annotated output to the error function. Then we calculate the gradient of this error function in relation to the parameters of the network; using the backpropagation method to find the gradient greatly improves efficiency and numerical precision. We use the gradient to inform the network to make small changes in each of the values of its parameters in order to reduce the error output. In practice, training a network is a very intricate process that involves many "hyperparameters" not involved in the actual network but involved in training the network. These additional parameters are critical to training the network efficiently and minimizing problems such as overfitting.

Some additional issues with training a network involve the vanishing/exploding gradient problem and lowered training rates for initial layers. In practice, these make it difficult to train networks with more than a few layers without special network architectures which we will discuss later. We will not go into too many details here; however, we do want to make clear that much of the time spent on getting a network to work well are spent in tuning the training phase and running the backpropagation algorithm.

We would also like to note that a critical part of training the network is in partitioning all available data into a training set, a validation set, and a test set. We only train the network on the training data, while the other two sets are used to test the actual performance of the network on unseen data. Partitioning gives us a method to measure how well the network might perform on the unseen data the network sees in practice.

## 2.4 Vision in the mammalian brain

Now, in an effort to understand how neural networks may be applied to visual understanding, we consider the structure of the mammalian brain. Visual information enters through the eye and is focused by the lens onto the back of the retina. Here, it excites an array of photoreceptor cells, which send the visual information to the lateral geniculate nucleus (LGN) and onto the visual cortex of the brain. In the visual cortex, the information is processed, and the result is an understanding of our surroundings that enables us to take appropriate actions in pursuit of some objective. There are several interesting details in this process.

Prior to entering the visual cortex, the visual data concentrated on the retina are compressed to reduce the amount of information the brain needs to process.[18] The compression is enacted by the density of light-sensing cells on the back of the retina. Moreover, most of the information sent to the brain is concentrated at the fovea, located at the optical axis of the eye, with the concentration diminishing further away from the fovea. This concentration gradient is such that as objects in the periphery of your view get closer to your eye, almost no new information about the object is processed.[18]

From the retina, the information flows through the optic nerve to the LGN. The receptive field of the LGN is concentric with antagonistic regions. The antagonistic regions prevent diffuse light from stimulating the neurons. Neurons in the LGN filter for color and spatial patterns. Various neurons in the LGN also respond to ranges of temporal frequencies suggesting the role the LGN has in preprocessing motion in scenes.[18] In addition, the LGN can be demonstrated to have an important role in scene understanding and likely has a role in integrating information from other regions of the brain. It has been shown that in the absence of the primary visual

cortex, a subject is consciously blind; however, the LGN supplies enough information such that the subject can respond to visual cues they do not consciously recognize.[19] These studies suggest the integral role the LGN has in image processing.

Subsequently, the processed information is forwarded to the primary visual cortex (V1). Once the information reaches the visual cortex, it propagates through a series of hierarchical layers (V1, V2, V3, V4, and V5 among others) which continue to process the visual information for spatial and temporal relations.[18,20] As the visual information flows through the hierarchy, the work of low-level neurons is built up to form a hierarchical representation that builds upon itself and constructs abstract object recognition from low-level features. In addition, there are various regions in the brain not primarily associated with vision, like those that process other senses and those associated with motion, that have connections to and from the visual cortex.[20] The high level of interconnectivity between non-forward-flowing regions suggests the important relation of localized visual understanding with full scene understanding.

We can measure what level of recognition is occurring in a given region by analyzing the stimuli that trigger a given neuron. The set of light patterns that trigger an action potential of a neuron is called the receptive field of the neuron. Neurons in the visual cortex can, broadly, be broken into the category of simple or complex based on their receptive fields.[21]

Neurons with a simple receptive field have a roughly linear detection field. The receptive field may look something like a rectangular window where light within the window triggers an action potential. If two distinct stimuli individually stimulate the neuron, displaying both simultaneously will combine their effects in an approximately linear fashion. These simple receptive fields also have mutually antagonistic excitatory and inhibitory regions. If light shines in both the inhibitory and excitatory regions simultaneously, the stimuli's effect will be dulled.[21]

Neurons with a complex receptive field, however, are highly nonlinear to stimuli. For example, shining two stimuli together that, alone, each stimulate the neuron, may cause the neuron to have limited response. These neurons take as input activation potentials from neurons with simpler receptive fields and so can be thought of as a nonlinear composition of simpler filters.[18,21]

Neurons associated with binocular vision (i.e., those observed to be stimulated by both eyes) are affected in a summing manner. These neurons are observed in a low hierarchical layer, V1. The neuron responds in a similar manner when each eye is exposed to the stimulus individually. When both eyes are exposed, the neuron responds more strongly with the summed effect of both eyes. It is also possible for one eye to see an excitatory stimulus, and the other eye to see an inhibitory stimulus, resulting in a weak response from the neuron. In most neurons, a particular eye has a dominant response, although this dominant eye is not uniform to every neuron.[21]

From a psychological perspective, depth perception is studied as a degree of precision with which depth can be measured from various cues including binocular vision.[22] However, other cues such as image blur[23] have been demonstrated to contribute to depth perception. These suggest that the binocular stimulus seen early in the visual cortex does not alone give rise to depth perception but rather is integrated along with various other filtered features extracted in later layers.

Another notable feature of the visual cortex is that neurons with similarly oriented receptive fields are frequently organized in columns, each processing the output of the previous neuron in the column.[21] These columns, which are repeated to a high degree,[18] also tend to feature interconnections.[24]

Although the majority of information is transmitted up the hierarchy, from simple to complex, there are a significant number of backwards and horizontal connections down and across the hierarchy. These connections, rather than stimulate neurons, are seen to modulate the processing performed by the neurons.[24] This feedback and additional input to visual processing suggest the importance both of temporal and external understanding for visual understanding.

After traversing through the visual cortex layers, two of the key regions visual information is integrated in are the middle temporal cortex (MT) and the inferior temporal cortex (IT). Especially interesting about the MT region is that it integrates information from throughout the hierarchy, while the IT generally receives information from only the level preceding it.[20,25] These relations suggest both the importance of hierarchical representation for visual understanding and the importance of low-level information for high-level tasks.

Now, we seek to extract key features of the mammalian visual understanding system with the intention of generalizing the approach to an automated system. An image is preprocessed in the retina and LGN before reaching the visual cortex. In the visual cortex, the visual information is processed in a semi-hierarchical manner, sees input from outside sources like those relating to other senses and memory, and finally, is integrated to produce understanding and responsive action in regions like the IT and MT. Then, abstracting the biological terminology, we have the following system: (1) preprocess image to fit various constraints like image size, (2a) extract information from the image using hierarchical processing and (2b) allow interconnections between nonvisual input sources and automatic modulation of processing through feedback and horizontal connection within a hierarchical level, and (3) integrate results from throughout the hierarchy to produce some output conclusion. The typical automated visual systems currently employed loosely follow steps 1, 2a, and 3.

## 2.5 Automated vision with convolutional neural networks

Computer vision is primarily concerned with recognizing the low-dimensional features and understanding from a comparatively high-dimensional image space. Here, we mean that a

small color image could have several hundred thousand pixels, each pixel consisting of three color channels. However, there may only be a few pieces of information a human picks out: a chair, a desk cluttered with paperwork, a bed, and evening lighting in a bedroom. This drastic reduction in the size of the space (from several hundred thousand pixels to several scene features) is a key reason why CNNs make sense on image data. The convolution operation is fundamentally geared to reducing the size of the input data.

A CNN in its simplest form is nearly identical to the deep neural networks we introduced earlier. The primary difference is the use of "convolutional layers" as opposed to fully connected layers in the initial layers of the network. In each convolutional layer, we apply a series of filters; the number of filters is the choice of the network's designer. Each filter is a small matrix, typically smaller than $11 \times 11 \times D$ (D is the depth of the input volume; three for a starting RGB image), of variable values learned by the network. This matrix is applied as a sort of sliding window to each patch of the input, with piecewise multiplication and subsequent addition over all the values. This operation amounts to convolution. The size of each filter and its stride length are again parameters chosen by the networks.

Because each filter takes into account spatially local pixels in computing its output, it is an extremely useful operation for operating on images, which generally have a high degree of spatial relation between pixels (single objects are not disjoint). In addition, there is a large reduction in the number of learned parameters for the layer. Consider that a small 2D color image may have over $10^5$ input values; then a fully connected layer will have on the order of $10^{10}$ values to learn. This is far too many – the time to train the network will be too great, and moreover, will likely result in egregious overfitting. In contrast, a convolutional layer may involve on the order of $10^3$ learnable parameters: for example, a $3 \times 3$ filter has 9 and 100 filters in the layer which gives a total of 900 parameters. Intuitively, each filter allows us to learn something like an edge detector for a small image patch and apply this same edge detector throughout the image. The location of the edge in the image does not matter to the edge detector, so we can apply the same filter to every location in the image. Later layers of convolution apply a similar principle. They might detect compilations of specific types of edges in a specific orientation. The hierarchical view of images employed by the brain is represented through convolutional layers; in practice, it works very effectively.

There are several different strategies used in practice for image understanding with CNNs. The most simple is inputting an image, passing it through several convolution layers followed by a few fully connected layers at the end. The result of this method could be a highly optimized feature extractor for object recognition. Another common strategy is to create several lower resolution copies of the image through downsampling and passing each through a CNN. This multiscale approach allows the network to easily identify spatial relations on both low and high levels.

It is also worthwhile to mention deconvolutional neural networks here. Deconvolutional networks incorporate deconvolution and un-pooling layers, which seek to undo their counterparts in normal CNNs. When used, deconvolutional networks are typically attached to the end of a CNN and have a mirrored set of layers (e.g., a convolution in the CNN is mirrored by a deconvolution in the deconvolutional neural network). Although it is not possible to use exact inverses of the convolution and pooling operations by design, these operations "upsample" their inputs to produce outputs with larger size in a manner that reflects the operation they are undoing. Deconvolution layers perform convolutions with weights typically determined by their corresponding convolution layer; their inputs are padded such that the output of the layer is larger than its input. Un-pooling layers typically require a network to keep track of which pixel was picked from the corresponding pooling layer and use this knowledge to upsample the input. These upsampling architectures are generally used to reconstruct an image or else learn to generate image data.

## 2.6 Additional deep learning networks

Although the focus of this review is 3D computer vision, we find it is necessary to discuss some network architectures that have seen greater application in areas like automatic translation, text generation, and other categories. In many applications, it is fruitful to utilize more than one type of network; for example, the task of automatically captioning images.[26] Additionally, we believe these other types of networks are a good source of inspiration for future advances in computer vision and the extension of related fields.

One that has seen application to computer vision is the deep residual neural network.[13] It is an extension of a deep neural network that connects a previous layer's output to a subsequent layer's output by summing them. Rather than having the subsequent layer learn to output an independent value given its input, it will learn to output a (small) residual change to a previous output. Intuitively, this should be easier to learn, and in practice, it allows one to construct and train very deep networks with over a hundred layers. An extension to the residual network developed for image processing is the dilated residual network.[27] This network offers an improvement from the residual network by modifying the convolution operation to increase the resolution of inputs in deeper layers of the network. Intuitively, this will better preserve spatial relations between the pixels in the deeper layers of the network and allow the network to use these spatial relations throughout the network.

Recurrent neural networks (RNN)[28] are useful for temporally dependent data inputs. A typical use is for textual understanding. The idea is to have a regular network (which may or may not be deep) with special memory unit layers. These layers self-modify in some learned manner after each input. One of the most well-known types of memory units is the long short-term memory unit (LSTM).[29,30] There are a number of variants of the LSTM with slightly different functionalities but

generally similar structures. To illustrate how an RNN might work, consider the context of textual understanding where the RNN is processing a sentence. The network the second word in a sentence sees will be modified from the network the first word sees. Importantly, it is this first word that determines the modification for the second word. Then the network the last word sees is a result of all previous words in the sentence.

Next, we discuss two network architectures that can be used to create new data. In a generative adversarial network (GAN),[31] we train two networks against each other. The first network, the generator, attempts to generate new, realistic data. The second network, the discriminator, seeks to discriminate between the generated data and the real data. By setting the two networks against each other, it is possible to produce a very sophisticated classifier and detector from the first network, as well as the possibility to generate new data realistically with the second network.

Variational autoencoders (VAEs)[32] also allow the generation of new data. An input (typically an image) gets passed through the first half of the network. The data is downsampled through typical layers of a CNN, and by the middle portion, the entire input has been encoded as a small feature vector. The second half of the network is a deconvolutional neural network that reconstructs the original input from this feature vector. Of course, it cannot be an exact recovery because of the dimensionality reduction of the feature vector from the input. However, the reconstruction can capture key features of the data. In several applications, this allows one to modify the feature vector with the purpose of adding or removing specific, controlled features from the original data.

We will see the use of some of these networks in interesting applications of computer vision.

## 3 3D computer vision with deep learning

### 3.1 Introduction to 3D computer vision

Because of the wide availability of color cameras and the low availability of depth cameras in the past, much of the research in the computer vision setting has been performed for 2D images. However, with the recent availability of affordable, commercial 3D cameras (Microsoft Kinect, Intel RealSense, and others), it has become possible to work with real 3D data for computer vision tasks.[33] With the surge in interest in computer vision for use in a wide variety of areas including autonomous robot control, virtual reality, and medical-related tasks, there has also been a realization that depth information can provide a full 3D understanding for autonomous robots and dramatically increase accuracy in the object recognition, detection, and modeling tasks so critical to the previously mentioned applications.[34] In fact, some applications that require the ability to accurately gauge distance or 3D object shapes like self-driving cars are dependent on 3D vision. Following the shift from traditional techniques to deep learning for 2D images, initial work in 3D vision sought to incorporate depth information into established 2D CNN frameworks with various small modifications. However, more recent work has realized the need to invent new CNN architectures and deep learning techniques to optimally gain a full 3D understanding.

There are various techniques used in commercial cameras to capture depth information. All of these techniques produce an image pair – one image is color, and one image is a depth map. To produce this pair, some depth cameras employ a pair of offset color cameras. The spatial disparity between the cameras allows an inference of the distance of objects in the scene from the camera. This stereoscopic method is very similar to our own binocular vision and provides moderate accuracy over a range of close and far distances. The other method involves a color camera along with an infrared camera and transmitter. The color camera captures the RGB information of a scene. The infrared transmitter produces a light pattern that is distorted and reflected upon hitting a surface. The infrared camera captures the distortion and reconstructs a depth map from here. Another depth imaging technique involves light detection and ranging (LIDAR), but it typically incurs higher costs. A LIDAR system emits pulses of light and measures their time to return to find distance. LIDAR offers high accuracy over very long ranges.

In the following sections, we will cover major advances and techniques used in 3D computer vision. To facilitate good discussion, we organize the field into the broad categories of classification, segmentation, and generation. We also separately summarize and discuss interesting datasets in use and their shortcomings, deep learning network architectures, and various techniques for depth representation.

### 3.2 Inputs to 3D computer vision networks

Although one may expect a 3D computer vision network to exclusively take as input a 3D volumetric map of a scene, there are a wide variety of ways to incorporate depth information into a vision system. We mention the most common inputs seen in practice, as well as the types of training data used as background for the following sections.

The simplest form of input to a 3D CNN is a 2D image. This is typically used in the context of inferring depth information from a 2D image or identifying 3D model information from 2D features. While the inputs are 2D, the CNN is typically trained using additional information from depth data or a 3D model.

Some networks also take a sequence of 2D images of a given scene. These multi-view systems seek to learn depth information through multiple images without explicitly calculating distance values. In practice, these techniques can work very well in areas like object recognition and 3D reconstruction. Part of the reason they work well is the abundance of 2D image training data, so we can expect their usefulness compared to 3D image data to diminish in the future.

The most common form of true 3D image is the color–depth pair known as a red-green-blue-depth (RGB-D) image. These images offer a partial 3D view into a scene, allowing a network to learn full spatial relations of objects. Some networks are designed to exclusively use depth information. Many times these networks are used in parallel with other networks that exclusively handle color information. It is also increasingly common to use 3D voxel grid inputs, with each voxel containing color information and depth information represented by the relative positioning of voxels. This representation, though a truly 3D representation of the scene (as opposed to the previous representations which involve 2D images and not 3D volumes), is often the most difficult to work with due to computational and memory constraints.

In addition to the variety of test-time inputs described earlier, there are many types of training inputs. Although some networks only see 2D images at test-time, additional annotated 3D information is generally required to train the network (e.g., the 3D information is used in the error function). In some cases, pure 2D information is used in a pre-training step. Subsequently, labeled 3D images are used to train the network, where the labels take the form of objects inside labeled bounding boxes, scene or pixel/voxel labels, or labeled common points between multiple images of a single scene. The specific type of labels used depends on the task. In general, it is common to use additional annotated data at training time to formulate an error function, while at test time the error function is not necessary and so this additional data is not necessary. In some applications with a goal of learning a 3D model, it is common to have artificial training data. Here, a 3D CAD model is constructed and labeled and sometimes used to generate a rendered scene with a graphics rendering engine. Although the model is not "real," it is often realistic enough that it can generalize to real inputs. Moreover, it is much easier to obtain exact information on object shapes and textures, lighting environments, and orientation of camera and objects. Depending on the desired behavior of the network, images can be sampled as 2D views or 3D views, while the full 3D map is available for ground–truth comparison. It is also becoming more common to use real 3D maps of scenes, although use is limited by low availability of accurate, rich, and annotated scenes.

## 3.3 Overview of training

To train the network, the standard approach used in 2D CNNs is typically used. We show the network an image, allow it to output a value, and then use backpropagation with a designed error function to update the network parameters and reduce error.

Because of the limited availability of annotated training data in a 3D setting, it is common to augment a training dataset to artificially increase its size. This technique can involve adding limited random noise to an image or voxel map, translating a 2D image in the case of a 3D rendered model, mirroring and reflecting an image, and other minor distortion techniques. The idea here is that, due to lack of training data, it is especially important that the network is able to generalize. Typically, there is no difference in the scene caused by translation or reflection or other forms of minor distortion. This data augmentation allows the network to train on more data while also reducing the danger of overfitting.

## 3.4 Classification and segmentation

Classification refers to several types of tasks designed to identify a specific object or scene. In a 2D setting, this could mean outputting a class for a picture of a single object or of a scene. We also include it to mean the detection of objects in a scene, performed typically through bounding boxes indicating approximate boundaries of an object. Finally, performing these two together gives a semantic segmentation of a scene, which involves detecting all objects and classifying them. In the 3D setting, classification takes a similar meaning with the additional challenge of recognizing the 3D shapes of objects, scenes, and the 3D relation of objects within scenes.

Segmentation is the classification of individual pixels. So, it can be thought of as fine-grained classification. A notable difference is that segmentation does not deal with the notion of individual objects, but rather which pixels in a picture belong to which category. Often, we are interested in segmenting pixels into foreground/background categories even rather than object classes. However, recent approaches have shown that classification and segmentation are highly interrelated problems, and that it is critical that they are handled together.

Prior to the use of CNNs for object classification,[35] common methods for classification involved hand-engineered feature detectors like the scale-invariant feature transform (SIFT) and the histogram of oriented gradients (HOG) followed by machine learning techniques like support vector machines. The idea here is to encode an image as a low-dimensional vector of real numbers with the feature detector. This representation is designed to be largely invariant to image modifications like rotation and scaling as well as robust to small variations from noise. The machine learning techniques then would be trained to separate the low-dimensional vectors into their respective categories, allowing for classification.

CNNs are able to learn the feature detector task; although they result in features with many similar properties as SIFT and HOG, these features are more optimized to the given data. Moreover, it is much easier to train a CNN if you have enough data than to manually design a feature detector. Similarly, image segmentation tasks were typically handled with specialized algorithms designed to discriminate similar patches of color, pattern, or other features. Initial work focused more on using CNNs for classification while using the engineered algorithms for segmentation, but more recent work is also using CNNs for segmentation.

In an early attempt at using a CNN on RGB-D data, Couprie *et al.*[36] use a multiscale CNN for semantic segmentation of objects from a color–depth pair of images. The network involves two parallel paths; the first is a CNN tasked

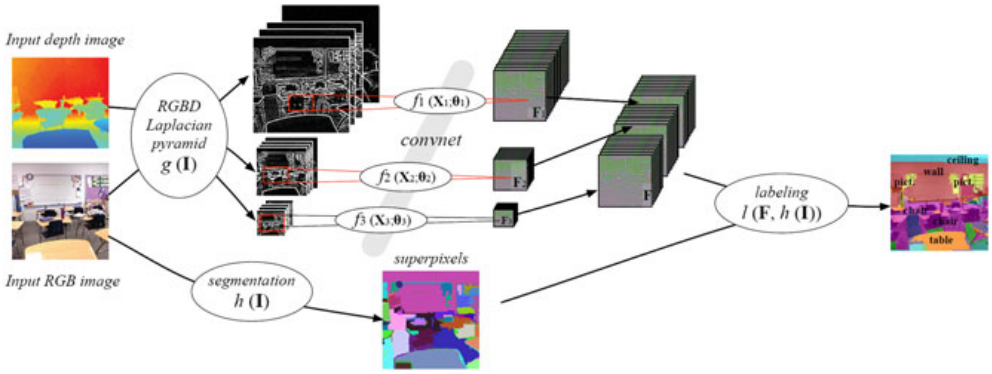



**FIGURE 1** — Network model for learning semantic segmentation. [Reprinted with permission][36]

with classifying objects in the scene, while the second is a function that segments the scene (Fig. 1). Only the classification network uses the depth information. Prior to inputting into the network, the RGB-D image is preprocessed to force zero mean and unit standard deviation in localized pixel neighborhoods. The depth information is added as a 4th channel to the three RGB channels of the image, and this four layer image is input into a CNN.

For purposes of a comparison, the authors trained a second network that only used 2D color images for classification. The accuracy of the semantic segmentation varies with the type of class involved. Objects with consistent depth information throughout the training data, like the floor and ceiling, saw improved performance; objects like books with non-consistent depth information from varying foreground/background positioning and shape saw a decrease in accuracy. On average, there was a minor improvement in classification accuracy of roughly 2% when using depth information.

We note several takeaways from this work. The first is the network's impressive performance compared with the then current state of the art, which it improves by 6% in average classification accuracy (4% when not using depth information). The authors demonstrated the applicability of a simple CNN architecture for combined use of RGB-D channels to perform better than hand-engineered feature detectors in a classification setting. The second is the lack of use of depth information for segmentation. As we will see later, depth information gives substantial improvements to image segmentation because of the clear separation of objects in 3D space. Lastly is the limited improvement from the additional depth information compared with just the RGB input. We see two clear reasons: the data set trained on is very limited in size, and the depth information was not used in the best possible representation. We get the second point from the variable results on specific classes. The CNN learns typical depth maps for given object classes but does not seem to get a sense of a 3D object. We will see later various depth representations that have been developed that perform significantly better, although they have their own drawbacks.

An alternate approach is suggested by Gupta *et al.*[37] In their work, they describe a system that uses a single RGB-D image to produce an object detection and segmentation system. As opposed to the work of Couprie *et al.*,[36] the authors endeavor to use the RGB and depth images separately.

Following a non-neural network based contour detection and region proposal scheme extended for RGB-D data, they use two separate CNNs for feature extraction (Fig. 2). Both CNNs are based on the architecture used by Krizhevsky *et al.*,[35] trained with ImageNet[38] data, and fine-tuned on the given datasets. The first CNN is for the RGB image, while the second CNN is for the depth data. Rather than using the raw depth data, the authors introduce a novel geocentric encoding with three features: horizontal disparity, height

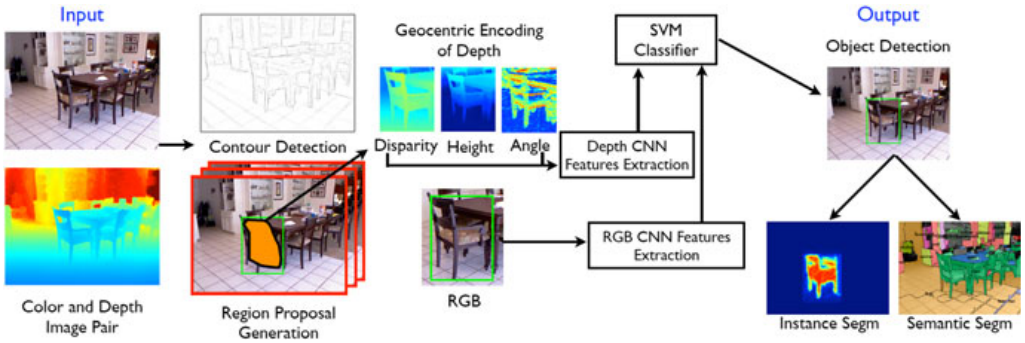



**FIGURE 2** — Pipeline for object detection and segmentation system. [Reprinted with permission][37]

above ground, and angle relative to gravity. Interestingly, they show that pre-training on ImageNet data enables better results for the depth feature extractor. This result implies that the depth-based features encode information in a similar manner as color images.

The main points of interest to us are the separation of depth and color channels, the use of nearly identical CNNs for both RGB and depth feature extractors, and the use of extrapolated features in place of a depth map. Although it is difficult to compare with the work of Couprie *et al*.[36] to see the effects of each individual change, the authors do see significant improvements. The later works we look at use the more modern and computationally expensive approach of voxel volumetric grids to represent 3D scenes rather than flat 2D images of color and depth.

In the work of Wu *et al*.,[39] a new network based on a convolutional deep belief network is proposed to more appositely model shape representation (Fig. 3). This network, called 3D ShapeNet, is trained to recognize RGB-D views of object models and performs various reconstructions that

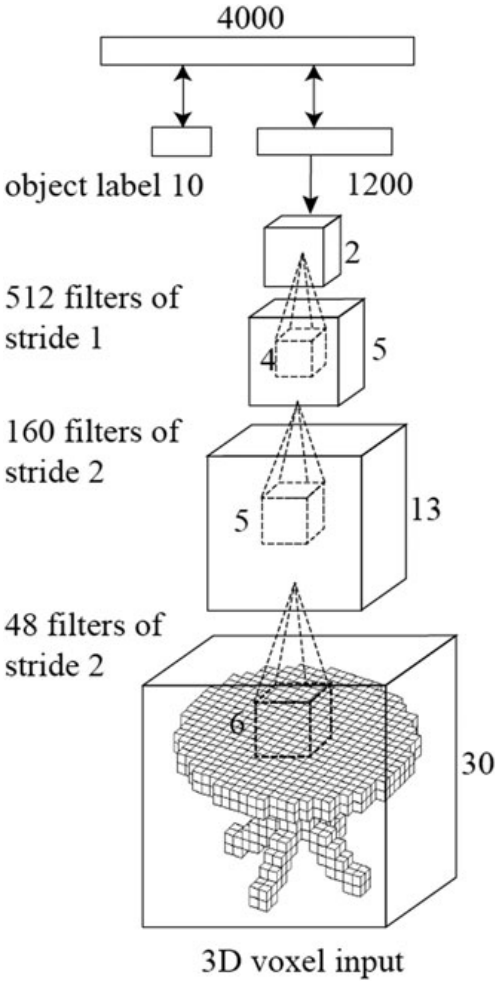



**FIGURE 3** — 3D convolutional neural network proposed by Wu et al. [Reprinted with permission][39]

we detail in the following section. The network represents objects as 3D probabilistic voxel grids.

The network performs 3D convolutions – rather than a 2D sliding window, a 3D sliding box is used. In addition, atypical from standard CNN models, 3D ShapeNet does not use pooling layers. Note also the limitations imposed by a 3D voxel model: the additional dimension prohibits high-resolution models due to computational intractability. Here, the voxel grid is fixed to a size of $30 \times 30 \times 30$. The network is trained on isolated views of 3D CAD models without background clutter or additional objects.

During testing, the 3D ShapeNet is input a single depth map for classification. The depth map is transformed into a voxel grid representation where voxels are assigned values as surface, free space, or occluded signifying the value is unknown. Then, a classification is found through Gibbs sampling, which is used to estimate the occluded space and find an optimal label.

The authors offer two results for 3D ShapeNet when tested on real-world depth images from the NYUv2 dataset.[40] The first involves directly applying the network to the real-world images, and the second involves fine-tuning the network on the real-world images. In general, the fine-tuning improves the object classification performance, and overall, the network achieves a 57.9% classification rate on 10 class categories.

Although this result can certainly be improved upon, it does suggest pre-training on the CAD models is a viable option that absolves the need for high-fidelity labeled 3D data. Additionally, these results are rather impressive given the low-resolution voxel grids used for the network. The subsequent works we look at in this section continue the trend of using a full volumetric model for a 3D image.

Similarly, Milletari *et al*.[41] use a volumetric CNN with 3D convolutions to learn volumetric segmentation of prostates in magnetic resonance imaging data. They use a residual network architecture with convolutions followed by deconvolutions, with the final output being a proposed 3D segmentation. Due to limited training data, they also introduce random variations in training input to augment their dataset. The end result is competitive with the optimal published result. We find the network architecture used especially interesting, as the structure of bringing the data to a small subspace through convolutions, and then upsampling to the original sized input space through deconvolutions is common in networks used to generate features.

In their recent work, Song and Xiao[42] develop a 3D CNN for object detection from a single RGB-D image of a scene. The CNN, called a 3D region proposal network (RPN), produces 3D bounding boxes of objects given a single RGB-D image of a scene (Fig. 4). They also propose a joint object recognition network, which uses both the 3D view and the 2D color image to separately identify feature vectors for the detected object, before concatenating the feature vectors to learn object classification and a regression for the 3D object box (Fig. 5).

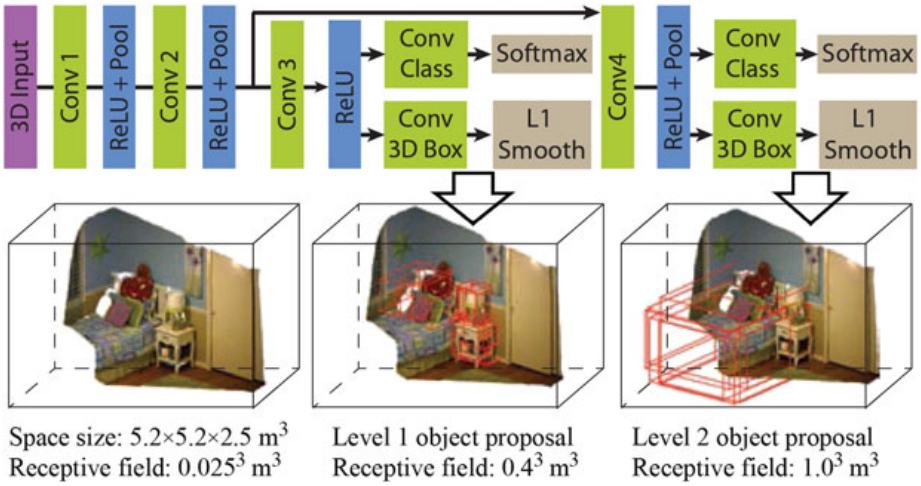



**FIGURE 4** — Convolutional neural network for object proposal regions at two object size levels. [Reprinted with permission][42]

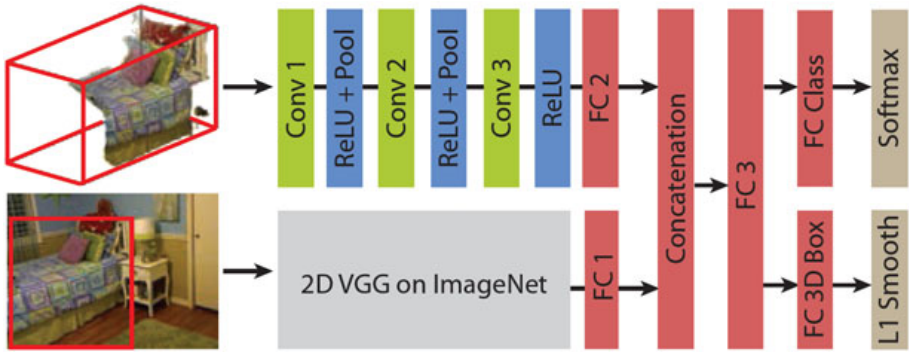



**FIGURE 5** — Joint 3D and 2D convolutional neural network for object recognition. [Reprinted with permission][42]

The proposed RPN uses a multiscale approach, where two different sizes of the 3D scene are used. This allows each network to focus on object detection at two different scales. The RGB-D view is represented as a 3D voxel map with a resolution of 208 × 208 × 100. Depth is encoded using a truncated signed distance function (TSDF) that gives the distance to the nearest surface in three directions (e.g., the x, y, and z directions). This encoding splits up the depth map into a 3D voxel grid where values are assigned based on distance to a surface; sign is determined by whether the voxel is occluded. Using TSDF gives a better sense of the geometry of the scene as opposed to a raw depth map. The RGB values can be appended to each voxel, enabling a compact 3D representation.

The RPN uses a set of predefined box sizes based on the statistics of objects in the images to find potential object fits in the scene. The scene is preprocessed to obtain information about its orientation; this allows for less ambiguity in bounding box orientation. Interestingly, the authors use the specifications of common depth cameras to define actual distances (meters) rather than pixel sizes for the voxel grid and bounding box sizes. This decision is intended to reflect real-world sizes that the network is attempting to detect.

The object recognition network is an especially interesting network that can be seen to work well. By allowing a 2D and 3D network to work together, it is possible to extract more information. The 3D network is limited by both training data and computation time – it operates on a 30 × 30 × 30 voxel grid of depth information. The 2D network uses the VGG network[43] along with some added layers. It is important to note that the VGG network is pre-trained on ImageNet data, which allows the overall object recognition network to leverage additional training data available through ImageNet. This joint network then allows for the use of additional depth data in a setting where the RGB-D data must be low resolution to be computationally feasible and where there is limited RGB-D training data. The authors compare this joint approach with a different network where they encode color directly to the 3D voxels rather than use the VGG network. The second approach does not work as well, likely for two major reasons. First, the voxel grid is low resolution, and much of the color information is lost in the voxel grid. Second, ImageNet is a significantly larger dataset than the ones available for 3D tasks.

The networks are trained and tested on the NYUv2 and SUN RGB-D[44] datasets. When tested for object detection,

an average of 84.9% out of 19 object classes are detected on the NYUv2 dataset using the 3D RPN. Interestingly, the color information has little impact, suggesting the trained 3D network primarily distinguishes using depth cues. As compared with a hand-engineered baseline for 3D detection, this deep learning approach offers an improvement of 14.4%. Additionally, the authors offer a comparison with a method that only uses the 2D data to find object region proposals; the method that uses only 2D data is far worse given its inability to detect partially occluded objects.

Here, we see a significantly improved use of 3D CNNs for object detection tasks. Although a joint approach is used where a 3D and 2D net are used in conjunction, there is reason to believe that a full 3D approach will perform better – the addition of 3D data improves 3D object detection compared with 2D object detection already. So the main challenges involve better representation of the 3D scene to enable higher resolution models. Of course, more data will also help. Lastly, we note the speed of the network to propose object locations. Although it is much faster than previous methods, it takes roughly 20 seconds to fully process each image during testing. For any real-time detection system, this is too slow; for comparison, most humans easily identify objects in a scene within less than a second.[45] So we can infer that significant improvements in efficiency while maintaining and improving accuracy are certainly possible and necessary for understanding a 3D scene.

We also briefly mention the work in scene reconstruction and semantic labeling conducted by Song *et al*.[46]; we discuss it more fully in the following section. They combine the tasks of scene completion and semantic labeling into a single framework to attain improved results from previous works that took the two problems separately. They are able to draw a number of interesting conclusions; one of which is that simultaneous classification and reconstruction improve the results of both tasks.

What we take from this result is that the recognition of objects is vitally important to full scene understanding, and understanding other aspects of a scene helps classify the objects. Future work should attempt to combine various problems into single frameworks, as this joint learning approach will likely improve the results of the individual tasks.

## 3.5 Generation

In this section, we will consider work that seeks to predict additional image information rather than simply categorize a pixel, object, or scene. This includes inferring depth information from a 2D image, reconstructing hidden voxels in a 3D scene, and even generating novel 3D models of generic object categories.

In the first work we consider, Eigen *et al*.[47] use a CNN to predict depth maps from 2D color images. They use a multi-scale architecture to first predict a coarse depth map and subsequently refine this prediction. To predict the coarse depth map, they use a typical CNN network with five convolution/max-pooling layers followed by two fully connected layers; the last fully connected layer is sized appropriately to produce an output depth map image. The intent of the first five layers is to repeatedly shrink the output allowing for a predictor accounting for the image as a whole. To predict the final output, the authors inject the output of the coarse prediction to a second CNN which maintains the size of its output throughout. This second CNN uses local features in the image to refine the prediction, and so it would not make sense to reduce the size of the output at each step.

Most of the following work we look at will, rather than use a fully connected layer to upsample a feature vector to an appropriately sized output, use deconvolution layers and unmax-pooling. Here, the authors allow the network to learn how to upsample through the fully connected layer; given limited size of training data, this approach is likely not as good.

Later work attempts to predict full 3D models as opposed to just depth maps using voxel mappings and joint classification,[39] multiple 2D views of an object,[48,49] and using an ensemble of complimentary schemes.[50]

As discussed in the last section, Wu *et al*.[39] develop a voxel prediction network for the purpose of 3D model classification given a single RGB-D image of the model. The proposed network is designed for classification and uses randomization (through Gibbs sampling) to predict the occluded points as free space or filled. As the main purpose of the network is classification, the model reconstruction is only used as an auxiliary feature to help classify. The following works have more vested interest in reconstruction, with networks specifically designed to reconstruct; these approaches give better results.

In Häne *et al*.,[51] the authors propose a network model that predicts a 3D model given a single color image, a single depth image, or a partial volumetric representation. Their main contribution however is their output voxel mapping. Rather than predict a grid of fixed-size voxels, they allow for variable size voxels using octrees. This enables them to predict surface voxels at high resolution, while occluded and free space voxels are at low resolution (Fig. 6). Because of the computational constraints related to high-resolution 3D voxel grids, this formulation is critical to enable more refined 3D model predictions.

The authors design a CNN to predict the octree structure. First, the input is encoded into a small feature vector using a CNN. This feature vector is then passed through a 3D deconvolutional network which upsamples the feature vector to produce a volumetric model. Throughout the deconvolutional network, the feature representation is cropped appropriately to predict at higher resolution appropriate voxels.

This work gives a concrete method for an end-to-end system that predicts 3D models from single images at significantly higher resolution than previously possible ($256 \times 256 \times 256$ vs $32 \times 32 \times 32$). This type of model can be used in general to enhance other models of voxel prediction. Other approaches to be explored include explicit surface prediction as opposed to voxel occupancy grids.

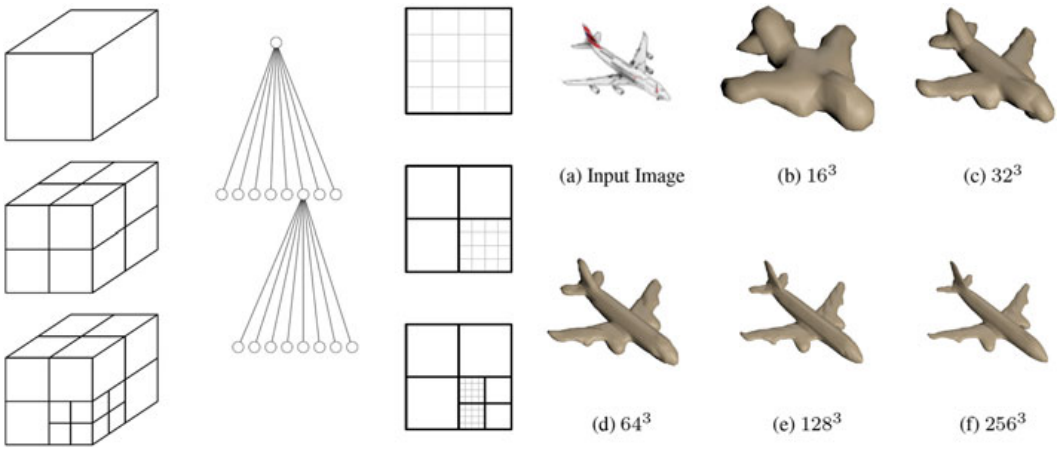



**FIGURE 6** — Left: an octree splits up a 3D volume into eight equally size octants. Right: the 3D model predicted for the plane can be seen to improve qualitatively at higher resolutions. [Reprinted with permission][51]

The authors in Choy *et al*.[48] create a novel network they name a 3D Recurrent Reconstruction Neural Network for use on both a single and multiple 2D images of a given object to predict 3D model (Fig. 7). The importance of this work is that it allows a single framework for variable amounts of input information; note that this would allow for online improvements to model prediction.

The network is a modified recurrent neural network. The encoder phase of the network is designed to compute a small feature vector representing the image. This is performed through a 2D CNN. The feature vector is then passed through a 3D convolutional LSTM (3D–LSTM), a novel unit developed by the authors. This unit is essentially a volumetric grid of regular LSTM units with limited connections. Each unit is responsible for a specific point of the reconstruction and takes inputs from a limited number of similarly localized inputs. The decoding phase is used to reconstruct a volumetric representation of the object. The decoding network features 3D convolution layers and un-pooling.

Impressively, this network is able to reconstruct realistic 3D models of an object given a single 2D picture in a cluttered scene (i.e., the object is not alone in the scene). However, there are some images where the prediction system fails, even on multiple images of the object. Given that the

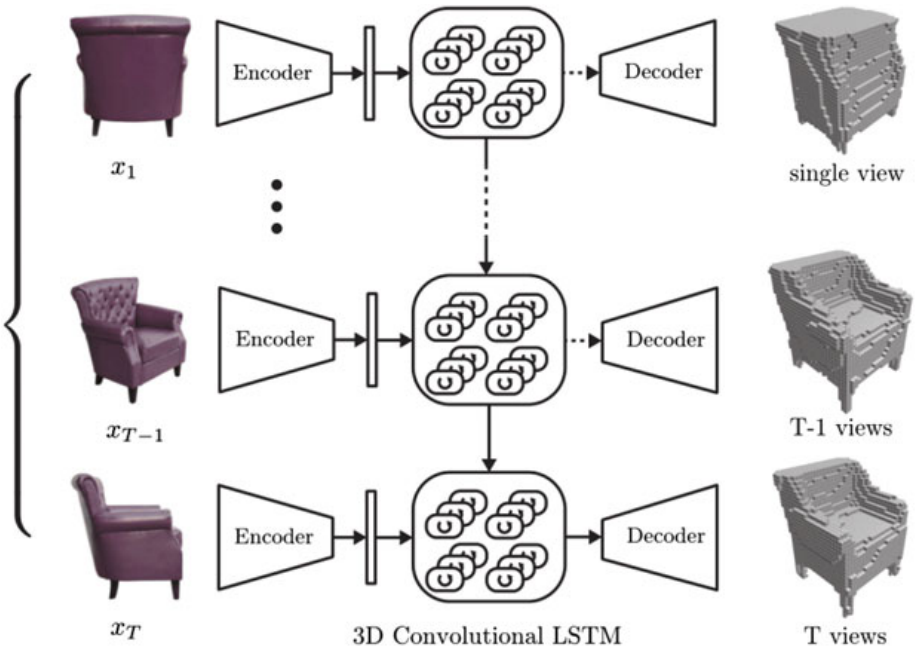



**FIGURE 7** — Top of the image shows network applied to a single image. Bottom of the image shows network applied to T images of the object, with the network's long short-term memory (LSTM) units updating each image. [Reprinted with permission][48]

network is not good at reconstructing features like thin chair legs, it may help to incorporate a multi-scale approach to improve reconstruction of fine-grained details. In addition, the limitations of 3D voxel representations are made apparent by the low resolution of the reconstructions: 32 × 32 × 32. In addition, we note that the proposed network has no knowledge of what the camera position change is. In many applications where the input images come from an autonomous robot with full knowledge of its movements, this formulation is too general. We can likely improve the precision of the 3D reconstruction if camera viewpoint information is included, so the network does not have to implicitly infer camera orientation. More work in this area needs to be carried out.

In Tulsiani *et al.*,[49] the authors use a modified error metric to train a CNN to predict 3D models from single view 2D images. For training data, they give the network access to one or more 2D views of the object to compute an error function based on the consistency of the predicted model with its multiple 2D views. To calculate consistency, they use a ray-tracing method that allows them to predict the most likely voxel placements for the observed views. The primary contribution of this paper is formulating rigorously a method for using these ideas to create a differentiable error function for use to train a neural network (which requires a differentiable error as it requires error gradients to update network parameters). The CNN is set up in an encoder–decoder format where initial 2D convolutions generate a small feature vector that is subsequently processed through 3D convolutions to generate the 3D voxel map.

There are a few obvious areas of improvement to this work. First, the resolution of the output voxel map is low as in Choy *et al.*[48] In addition, the CNN presented only allows for prediction from single views, while multiple views may be allowed in practical applications; therefore, adapting the neural network architecture proposed by Choy *et al.*[48] with the ideas from this paper may allow for better 3D reconstructions with multiple input images.

In Tulsiani *et al.*,[50] the authors create a pipeline for fully automated object detection and 3D model estimation from a single, real-world color image (Fig. 8). The pipeline involves several steps. First, the main object in the image is detected and then segmented using state-of-the art methods for joint detection and segmentation.[52] Then, the orientation of the object is inferred through a CNN. These outputs, along with learned deformable 3D models, are used to predict a coarse 3D model. This prediction is refined using an adaptation of the SIRFS technique to recover shape, illumination, and reflectance from shading.[53]

Notable about this scheme is its modularity and use of a variety of techniques. The object detection scheme involves region proposal through a multi-scale approach followed by feature extraction and classification through a CNN and a SVM and finally an additional refinement step to improve segmentation. The object orientation is inferred through a CNN. The 3D deformable models are learned by applying the expectation–maximization algorithm to class-specific 2D training data. The refinement is performed by using heuristic information about real objects (e.g., smoothness of object surfaces).

The system works reasonably well and especially so as the first fully integrated system for finding an object in a real-world RGB image and predicting its 3D model. However, we notice a few deficiencies. Although its modularity enables easy incorporation of improvements to individual parts, it prevents enhancements through more connectivity between parts (e.g., 3D model prediction informing object classification and segmentation). We would hope future research in

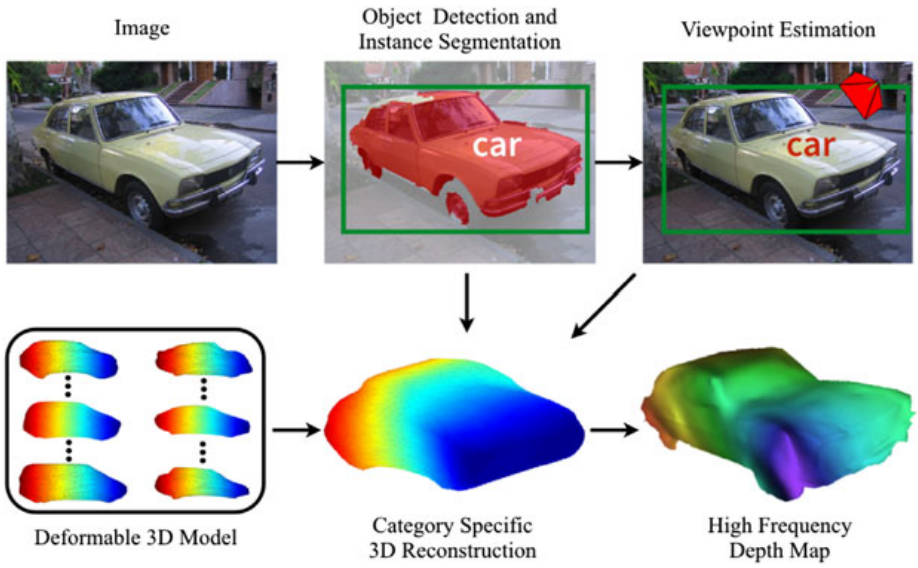



**FIGURE 8** — Pipeline to find object in a real-world RGB image and predict its 3D model. [Reprinted with permission][50]

this area of reconstruction attempts to create an end-to-end system to jointly learn all the aspects of the pipeline together. Although this approach is challenging, we believe it will improve accuracy of all predicted features due to each predicted feature informing the others. In addition, the proposed pipeline lacks the ability to integrate multiple views of the object to inform model reconstruction. This can easily be incorporated using existing approaches like that of Choy *et al.*[48]

In Song *et al.*,[46] the authors attempt to generalize object model prediction to entire scene completion for indoor scenes. Given a depth image, they develop a novel neural network, the semantic scene completion network, to both predict the obstructed voxels and assign a semantic label to each voxel (Fig. 9).

The 2D depth image is first encoded as a 3D voxel grid. Each voxel is assigned a value using TSDF, although the authors make some modifications to the generic formulation. This voxel grid is fed into a 3D CNN. To incorporate various sized receptive fields into the final predictions, features from throughout the network are concatenated near the end of the network. This is a sort of multi-scale approach and allows the model to predict using global and local features.

The authors also provide evidence for better prediction results due to the joint prediction of voxel class and voxel occupancy. As we have seen before, joint prediction of various features gives better results.

The authors also introduce a new synthetic dataset called SUNCG. This dataset contains realistic indoor scenes of homes. They use this dataset to pre-train semantic scene completion network and then fine tune on real-world data, noting >10% improvement when they use the synthetic data. Although more real-world data would likely improve their results, their results suggest synthetic data are viable substitutes.

Obvious extensions to this work are to allow multiple observations and to incorporate color cues for more refined voxel prediction. It would also be interesting to modify the prediction to, rather than give voxel predictions throughout the 3D space, only predict surface voxels similar to Häne et al.[51] This formulation would aim to reduce computation and allow for higher resolution predictions. It would also make sense to extend this model to outdoor spaces. This may be a bit more challenging, given the much larger scales typical of outdoor spaces. We may want the model to predict voxels at a variable refinement scale; close to the camera, we would want high resolution, and far from the camera, we can settle for lower resolution.

There is also some interesting research on 3D deep networks to learn object-like properties. One paper details a network trained to emulate a graphics engine allowing for modification of lighting and other properties in a scene.[54] There is also work that uses a GAN to generate novel 3D object models in a fully learned approach.[55]

In Kulkarni *et al.*,[54] the authors describe the deep convolutional inverse graphics network for use as a 3D graphics rendering engine. This network is essentially a modified VAE. The network uses a CNN to represent an input image as a feature vector and subsequently performs deconvolutions to reconstruct the original image. By modifying the feature vector between the encoder and decoder, it is possible to adjust various rendering parameters (e.g., angle of lighting). The key contribution of the authors is a method to train the encoder network to produce a disentangled feature vector for graphics rendering parameters. By disentangled, we mean that each value in the feature vector should more-or-less represent a specific rendering parameter.

In order to train the network to produce this feature vector, the authors use mini-batches which contain images with only a single parameter being modified (e.g., the angle of lighting is the only difference between each of the images). The network is allowed to compute a feature vector for each of these images. The desired output has all the feature vectors identical except in a single value used to represent the varied parameter for generating the images. To train the encoder network toward this representation, the authors propose temporarily replacing all values in the feature vectors

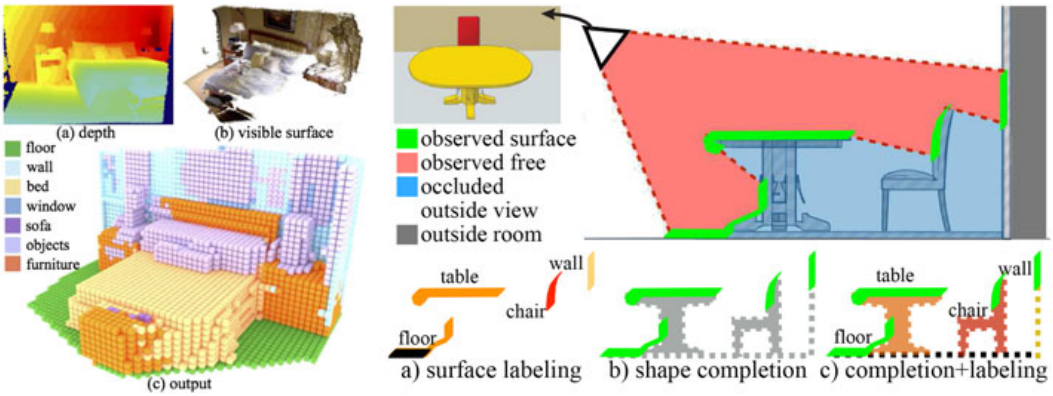



**FIGURE 9** — Left: given the depth image (a), SSCNet predicts the voxel grid (c) which fills in missing data and labels each voxel with a predicted class. (b) shows which voxels are visible and which are missing. Right: a visualization of the goal of SSCNet. SSCNet combines classification of voxels (a) and prediction of missing voxels (b) so that its output (c) both fills in missing data and classifies all output voxels. [Reprinted with permission][46]

with their average across the mini-batch. Only the value intended to change across image inputs is allowed to remain unchanged. The feature vectors are then passed through the decoder network, and error gradients are calculated for backpropagation. The authors also propose modifying the gradients to encourage the network to be invariant across those values that represent the non-modified parameters. By training the network in this manner, the authors produce a network that encodes images with an understandable representation that can be modified to produce controlled variations in the output image.

The current work is restricted in that it requires very domain specific training, with pre-segmented object depictions and grouping by single variations of parameters. This type of training data is not typically available and prevents generalizations to whole scene generations. However, we view this work as an important demonstration of the ability to push a network to a certain feature representation during training. Without explicitly modifying the network, the authors are able to attain a certain, useful representation.

Similarly, the authors in Wu *et al.*[55] use a generative network; however, they use a GAN rather than a VAE. The authors are able to create a network named 3D-GAN, which uses 3D CNN architecture combined with the GAN framework to produce novel realistic 3D models of objects like chairs or cars. Using a GAN allows for unsupervised learning, meaning that the training data does not have to be annotated. In addition, the GAN learns an intuitive representation of the 3D space where something like the chairs on an arm can be "added" to an armless chair by adding two low-dimensional feature vectors that are used as inputs to the generator part of the GAN.

The GAN allows for generating realistic object shapes by sampling feature vectors without any input required. Taken alone, the discriminator network of the GAN can be adapted for image classification. Despite limited training data, the network performs remarkably well on 3D model classification and achieves accuracy close to that of a supervised network.

In addition, the authors propose an extension of the 3D-GAN where they also incorporate a VAE. This network uses an encoder to featurize an input image, a shared decoder/generator, and a discriminator. This extension allows the network to produce a feature representation of an input image and subsequently use the GAN architecture to reconstruct a 3D model from the image.

Not much work incorporates GANs in 3D computer vision. As they allow for unsupervised learning on limited training data, increased incorporation of these types of networks could be immensely beneficial.

## 3.6 Datasets

In the previous sections, we observed numerous deep learning techniques for 3D computer vision tasks. Each of these techniques requires the use of training data; here, we summarize the most recently released datasets and their shortcomings.

There are two major categories for data used in 3D computer vision – images taken from the real world, and synthetic image data rendered from CAD models. While it is reasonable to assume real-world data is preferable, it is often prohibitively expensive to collect and annotate large quantities of such data. Additionally, it is difficult to get highly accurate ground truth 3D representations. In contrast, CAD models provide a (nearly) limitless quantity of exactly annotated data. Their primary drawback is that they do not have the exact statistics of the real images used for testing. However, synthetic data has seen highly effective use for pre-training. This use suggests that synthetic data is good enough to learn the general structure of real-world data.

We first consider 2D data. ImageNet[38] contains in excess of 1 M images with 1 k classes annotated with bounding boxes for object detection and labeled for object classification. Although this dataset contains only color images, it has seen use in the 3D setting due to its size and variety of classes. It can be useful for (pre) training a network that operates on a 2D image in conjunction with other networks that account for depth information. In more recent works that represent depth information purely with voxel grids, ImageNet is less applicable.

For 3D data, most datasets are smaller than ImageNet. However, there are some new datasets that rival ImageNet for size. Their size is especially important for learning high resolution 3D models given that 3D objects require more data to learn due to their extra dimension.

ModelNet[39] contains annotated 3D CAD models of 660 distinct categories, with roughly 150 k models in total. This data is useful purely for learning 3D object shapes. Given that it does not contain color information and only a coarse volume representation, it would primarily be useful for pre-training. SUNCG[46] contains roughly 400 k full room models, each manually validated for being realistic and filled with labeled object models. Although this dataset is also synthetic, the scenes it depicts are realistic. This is critical for learning object–scene relationships and enables networks to learn important representations from purely synthetic data. However, it is still beneficial to fine-tune with real-world data. SceneNet RGB-D[56] also uses synthetic data from indoor rooms and contains roughly 5 M highly realistic images; an unlimited number can be generated. However, scene layout is randomly sampled from a distribution intended to reflect reality. Although in practice, some of the generated images are of non-realistic scenes, the overall approach of using highly realistic synthetic training data may be a better approach compared with hand annotated real-world scenes; in addition, SceneNet RGB-D will be useful for pre-training.

ScanNet,[57] in contrast, features a very richly annotated dataset with semantic segmentation, camera orientation, and full 3D scene information composed from video sequences taken in real indoor scenes. Impressively, the dataset includes 2.5 M individual views. Although the ground truths provide by

synthetic datasets may be more accurate, ScanNet offers data that can be used to fine-tune a network on real images. Impressively, its large size may allow for it to be used without pre-training on another dataset.

Although recent 3D datasets have impressive size and detailed annotations, we note that they lack in variety. In particular, there is little data with outdoor scenes. Moreover, this type of data is more difficult to deal with. While indoor scenes occur in relatively small volumes, outdoor scenes can depict very large distances. Real-world outdoor data will also have significantly more noise. It is more difficult to collect and annotate real-world outdoor data given the wider variety of object categories and models.

## 3.7 Depth representation

Depth representation can be seen to be a critical modeling element of a 3D vision system. Here, we summarize the various proposals and their tradeoffs.

Initial attempts for deep learning in 3D vision sought to treat depth as another 2D image feature either through direct stacking as a 4th pixel feature[36] or through training a network to operate on 2D depth images enhanced with geocentric encodings.[37] These methods showed clear improvements to using just RGB images; moreover, they allowed for the use of ImageNet data for training. Given that there was little 3D training data available, this representation made sense at the time.

Later work shifted to volumetric representation of depth data either through 3D voxel grids representing 3D object localization in a scene or through 3D volumes created from depth maps like the TSDF used in Song *et al*.[46] These representations give a more fundamental view of scene understanding. Rather than treating depth as a 2D mapping, we treat objects as full 3D objects in a 3D scene. With more training data available, these methods have been able to perform significantly better. However, their main drawback is a low resolution because of prohibitively high computational cost from predicting every voxel in a 3D space. A recent improvement uses a hierarchical volumetric model that attempts to predict only those voxels at the surface of an object.[51] This allows for significantly finer resolutions.

In some cases, however, multiple 2D views are used as an implicit representation of depth. There are various tradeoffs between a single volumetric view and multiple 2D views (multi-view). These approaches are compared in Qi *et al*.[58] Multi-view is seen to perform better, although this is likely due to more available 2D data. With the recently released large 3D datasets, the volumetric convolution architectures will likely perform better.

Clearly, more work needs to be carried out to find the best methods for representing depth. The current models are well suited for operating on indoor scenes, but outdoor scenes consist of objects in varying spatial contexts: the foreground could contain objects less than a meter apart, while background objects could be kilometers apart. In addition, even the current models for indoor scene representation can be improved for better efficiency. While the hierarchical approach looks promising, other approaches like explicit representation of object surfaces should be explored.

## 3.8 Network models

Here, we summarize the various network models that have been proposed for 3D vision understanding.

Most network models involve CNNs of some sort, given that the input to the networks are images whether 2D or 3D. When operating on 3D volumes, the convolution operation is generalized to an additional dimension, while the architecture can largely be kept the same. The general trend has been to use as deep a network as is possible to train. Deeper networks tend to increase performance as they extend the hierarchical representation given by a CNN.

While CNNs are so popular because they reduce data size in a manner that incorporates key image descriptors, the low-dimensional output representations obscure fine-grained image features like the thin legs of a chair. A common workaround involves multi-level approaches that seek to incorporate both low-grained and high-grained features, typically through multiple CNNs each designed to generate features at a specific receptive field size. One approach runs multiple networks given downsampled versions of the input image and concatenates the features together.[47] Another approach takes features from multiple points in a single-path network and concatenates these features into hypercolumns.[59] Interestingly, these approaches mimic the visual cortex in some regions of the brain like the MT, which integrates visual information across a hierarchy.[20] The results of these studies suggest that a network learns useful information throughout its hierarchical representation, which can be used to improve the results in a variety of contexts.

There has also been work on representing variable sizes of input using the memory unit of an RNN. In Choy *et al*.,[48] the authors suggest a convolutional encoder and decoder setup, with a special memory unit sandwiched in between. This unit allows for the network to update itself using previous views of a scene. However, the suggested approach has limited generalization. Inputting more views during testing improves accuracy until a threshold determined by the maximum number of views of a scene given during training. Clearly, this type of online approach is important for any real-time system as it provides a nice framework for continuous improvement to any predicted model. We hope future research examines both the use of memory units and alternative approaches to this problem.

Other networks more concerned with generating models use VAEs and GANs. VAEs allow for encoding an intermediate feature vector that can be modified before reconstructing an output. In Kulkarni *et al*.,[54] this quality is used to create a network that behaves like a rendering engine. GANs allow for unsupervised learning and require far less training data than all the supervised networks we previously discussed. The

authors in Wu *et al*.[55] use a GAN-based architecture. After training, the generator network is able to generate realistic object models, and the discriminator network is easily adapted for classification tasks with near state-of-the art results (state-of-the art for unsupervised methods). The authors also propose using an architecture that combines a VAE and GAN for the task of model reconstruction given partial information (e.g., a color image). We strongly believe more research in unsupervised learning will be critical to future advances. Strong unsupervised learning methods remove the heavy dependence on richly annotated training data, which currently serves as one of the major bottlenecks of the field.

We also note that many architectures seek to improve upon previous results by incorporating multiple tasks into a single framework. For example, the authors in Song *et al*.[46] design a network to jointly classify voxels and predict occluded voxels, noting better results for both tasks. When performed correctly, joint prediction should always give better predictions – we are considering more information to better restrict the possible outputs.

The major benefit of a CNN compared with hand-engineered models is the end-to-end solution it provides. For simpler tasks like image classification, this works extremely well. However, for more complex tasks like detection and prediction of 3D model from a cluttered color image,[50] it seems that most researchers accept the need for more fine-grained control of the data path for predictions. Rather than relying on a generic CNN, these tasks often require modular system parts for different, specific tasks. Some of these parts are fully learned through a neural network, while others use heuristics and hand-engineering. The method for connecting these parts is also often a sort of hyperparameter of the network. While the results of these approaches are impressive, the network architectures are attuned to specific tasks and data and have limited generalizability. We believe that more research in joint learning of related tasks is critical in both discovering more generalizable network architectures and in improving accuracy of the individual tasks.

Lastly, we note that the development of techniques to increase the speed of learning and prediction is also vitally important. For any real-time visual understanding system, prediction speed is crucial. For any system in general, fast learning is necessary.

## 4 Looking forward

3D computer vision is currently at a point where tasks like object recognition in an indoor scene or realistic reconstruction of occluded objects can be performed with acceptable accuracy. We have elaborated on the current challenges and potential directions of future research. Here, we conclude with a discussion of potential areas of future research and their applications.

Perhaps, the most obvious extension to 3D computer vision is 4D computer vision, where we train on 3D video sequences rather than still images. The idea here would be to learn action associations in addition to (improved) object models. A related extension is the use of additional sensory inputs to inform visual predictions. Improvements in automatically learning relations between inputs will be critical for applications like 3D mapping, where we use a camera input to stitch together a robust 3D environment map while keeping track of the camera's location in the environment. In other applications, this type of approach could allow an automated system to attain, for example, both visual and auditory understanding as well as their interrelations.

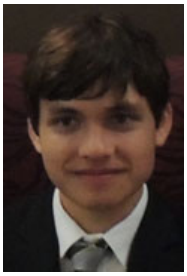
Of course, more quantity and variation of training data are also highly important, and innovations in collecting and annotating this data are needed. At the same time, advances in unsupervised learning and transfer learning (transferring "knowledge" learned in one domain or dataset to another) could alleviate the need for much of this training data.

However, we believe one of the most critical developments in 3D visual understanding will be a more unified end-to-end system for visual understanding. Although CNNs were the revolutionizing technique that ended much of the hand-engineering for tasks like 2D classification, they currently allow only modular understanding in common 3D (and 2D) vision tasks. For single tasks, they work fine. For multiple interrelated tasks intended to be joined together in a nontrivial way, most researchers have not sought to adapt a single network. Clearly, it must be possible for the entire framework (subtasks and how they relate to the overall task) to be learned – biological systems are capable of it. And so we hope future research strives for a more systematic approach. In the same way that CNNs perform far better as feature extractors than hand-engineered attempts, we believe learned frameworks will perform and generalize far better than hand-engineered systems.

## References


1 S. Thrun *et al*., "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, **23**, No. 9, 661–692 (2006).
2 D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, **521**, No. 7553, 460–466 (2015).
3 G. Hinton *et al*., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, **29**, No. 6, 82–97 (2012).
4 I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in Neural Information Processing Systems 27, 2014.
5 A. K. Bhowmik, Interactive displays: Natural human-interface technologies. Wiley, (2014).
6 R. F. Murphy, "An active role for machine learning in drug development," *Nat. Chem. Biol.*, **7**, No. 6, 327–330 (2011).
7 P. Sajda, "Machine learning for detection and diagnosis of disease," *Annu. Rev. Biomed. Eng.*, **8**, 537–565 (2006).
8 K.-S. Oh and K. Jung, "GPU implementation of neural networks," *Pattern Recognition*, **37**, No. 6, 1311–1314 (2004).
9 K. He, X. Zhang, S. Ren and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *The IEEE International Conference on Computer Vision*, 2015.

10 N. Srivastava *et al*., "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, **15**, No. 1, 1929–1958 (2014).
11 L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *International Conference on* Computational Statistics, 2010.
12 S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on* Machine Learning, 2015.
13 K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and* Pattern Recognition, 2016.
14 A. Ioannidou *et al*., "Deep learning advances in computer vision with 3D data: A survey," *ACM Computing Surveys*, **50**, No. 2, 20:1–20:38 (2017).
15 K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, **2**, 359–366 (1989).
16 R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," in *Conference on Learning* Theory, 2016.
17 Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard and L. D. Jackel, "Handwritten digit recognition with a backpropagation network," in Advances in neural information processing systems, 1990, pp. 396–404.
18 D. C. Van Essen, C. H. Anderson and D. J. Felleman, "Information processing in the primate visual system: An integrated systems perspective," *Science*, **255**, No. 5043, 419–423 (1992).
19 M. C. Schmid *et al*., "Blindsight depends on the lateral geniculate nucleus," *Nature*, **466**, No. 7304, 373–377 (2010).
20 D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cereb. Cortex*, **1**, No. 1, 1–47 (1991).
21 D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, **160**, No. 1, 106–154 (1962).
22 D. Vishwanath, "Toward a new theory of stereopsis," *Psychol. Rev.*, **121**, No. 2, 151–178 (2014).
23 M. Mauderer, S. Conte, M. A. Nacenta and D. Vishwanath, "Depth perception with gaze-contingent depth of field," in *Proceedings of the SIGCHI Conference on Human Factors in* Ceomputing Systems, 2014.
24 V. A. Lamme, H. Super and H. Spekreijse, "Feedforward, horizontal, and feedback processing in the visual cortex," *Curr. Opin. Neurobiol.*, **8**, No. 4, 529–535 (1998).
25 J. J. DiCarlo, D. Zoccolan and N. C. Rust, "How does the brain solve visual object recognition?" *Neuron*, **73**, No. 3, 415–434 (2012).
26 O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and tell: A neural image caption generator," in *The IEEE Conference on Computer Vision and* Pattern Recognition, 2015.
27 F. Yu, V. Koltun and T. Funkhouser, "Dilated residual networks," *arXiv preprint arXiv:1705.09914,* 2017.
28 Z. C. Lipton, J. Berkowitz and C. Elkan, "A critical review of recurrent neural networks," *arXiv preprint arXiv:1506.00019,* 2015.
29 S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, **9**, No. 8, 1735–1780 (1997).
30 F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to forget: Continual prediction with LSTM," in *International Conference on Artificial Neural Networks*, 1999.
31 I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," in Advances in Neural Information Processing Systems 27, 2014.
32 D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114,* 2013.
33 L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen and A. Bhowmik, "Intel RealSense stereoscopic depth cameras," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.
34 A. K. Bhowmik, "Interactive and immersive devices with perceptual computing technologies," *Molecular Crystals and Liquid Crystals*, **647**, No. 1, 329–340 (2017).
35 A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, 2012.
36 C. Couprie, C. Farabet, L. Najman and Y. LeCun, "Indoor semantic segmentation using depth information," *arXiv preprint arXiv:1301.3572,* 2013.
37 S. Gupta, R. Girshick, P. Arbeláez and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in *European Conference on Computer Vision*, 2014.
38 O. Russakovsky *et al*., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, **115**, No. 3, 211–252 (2015).
39 Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and* Pattern Recognition, 2015.
40 N. Silberman, D. Hoiem, P. Kohli and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *European Conference on Computer Vision*, 2012.
41 F. Milletari, N. Navab and S.-A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *3D Vision*, 2016.
42 S. Song and J. Xiao, "Deep sliding shapes for amodal 3D object detection in RGB-D images," in *IEEE Conference on Computer Vision and* Pattern Recognition, 2016.
43 K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning* Representations, 2014.
44 S. Song, S. P. Lichtenberg and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *The IEEE Conference on Computer Vision and* Pattern Recognition, 2015.
45 S. Thorpe, D. Fize and C. Marlot, "Speed of processing in the human visual system," *Nature*, **381**, No. 6582, 520–522 (1996).
46 S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva and T. Funkhouser, "Semantic scene completion from a single depth image," *arXiv preprint arXiv:1611.08974,* 2016.
47 D. Eigen, C. Puhrsch and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in Advances in Neural Information Processing Systems 27, 2014.
48 C. B. Choy, D. Xu, J. Gwak, K. Chen and S. Savarese, "3D–R2N2: A unified approach for single and multi-view 3D object reconstruction," in *European Conference on Computer Vision*, 2016.
49 S. Tulsiani, T. Zhou, A. A. Efros and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," *arXiv preprint arXiv:1704.06254,* 2017.
50 S. Tulsiani *et al*., "Learning category-specific deformable 3D models for object reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, **39**, No. 4, 719–731 (2017).
51 C. Häne, S. Tulsiani and J. Malik, "Hierarchical surface prediction for 3D object reconstruction," *arXiv preprint arXiv:1704.00710,* 2017.
52 B. Hariharan, P. Arbeláez, R. Girshick and J. Malik, "Simultaneous detection and segmentation," in *European Conference on Computer Vision*, 2014.
53 J. T. Barron and J. Malik, "Shape, illumination, and reflectance from shading," *IEEE Trans. Pattern Anal. Mach. Intell.*, **37**, No. 8, 1670–1687 (2015).
54 T. D. Kulkarni, W. F. Whitney, P. Kohli and J. Tenenbaum, "Deep convolutional inverse graphics network," in Advances in Neural Information Processing Systems 28, 2015.
55 J. Wu, C. Zhang, T. Xue, B. Freeman and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in Advances in Neural Information Processing Systems, 2016.
56 J. McCormac, A. Handa, S. Leutenegger and A. J. Davison, "SceneNet RGB-D: 5M photorealistic images of synthetic indoor trajectories with ground truth," *arXiv preprint arXiv:1612.05079,* 2016.
57 A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," *arXiv preprint arXiv:1702.04405,* 2017.
58 C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *IEEE Conference on Computer Vision and* Pattern Recognition, 2016.
59 B. Hariharan, P. Arbeláez, R. Girshick and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *IEEE Conference on Computer Vision and* Pattern Recognition, 2015.



**Kailas Vodrahalli** is currently a junior at UC Berkeley studying Electrical Engineering & Computer Science, expecting to graduate in 2019. He worked at Intel as an intern in the Perceptual Computing Group during the summer of 2017. His research interests include visual understanding, machine learning applied to learning frameworks, and medical electronics. He was an Intel STS Semi-finalist and gold medal winner at the international I-SWEEEP science fair in his senior year in high school.

**Dr Achin Bhowmik** is the chief technology officer and executive vice president of engineering at Starkey, the largest hearing technology company in the USA, a privately held business with more than 5000 employees and operations in more than 100 countries worldwide. In this role, he is responsible for leading the company's research and engineering efforts. Prior to joining Starkey, Dr Bhowmik was vice president and general manager of the Perceptual Computing Group at Intel Corporation. There, he was responsible for R&D, engineering, operations, and businesses in the areas of 3D sensing and interactive computing systems, computer vision and artificial intelligence, autonomous robots and drones, and immersive virtual and merged reality devices. Previously, he served as the chief of staff of the Personal Computing Group, Intel's largest business unit. Dr Bhowmik holds adjunct and guest professor positions, advises graduate research, and lectures on human–computer interactions and perceptual computing technologies at the Liquid Crystal Institute of the Kent State University, Kyung Hee University, Seoul, Indian Institute of Technology, Gandhinagar, Stanford University, and the University of California, Berkeley, where he is also on the board of advisors for the Fung Institute for Engineering Leadership. Dr Bhowmik was elected as a Fellow of the Society for Information Display (SID). He received the Industrial Distinguished Leader Award from the Asia-Pacific Signal and Information Processing Association. He serves on the executive board for SID and the board of directors for OpenCV. He has over 200 publications, including two books and 34 issued patents.