RGB-D Object Recognition using Deep Convolutional Neural Networks

by

Saman Zia

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

June 6, 2016

RGB-D Object Recognition using Deep Convolutional Neural Networks

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Saman Zia

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assoc. Prof. Yücel Yemez

_____

Assoc. Prof. Deniz Yüret

_____

Prof. Bülent Sankur

Date: _____

This thesis work was supported by TUBITAK under the project EEEAG-114E628

To the constants in my life; Mom, Dad and Hassan

ABSTRACT

Recent availability of low cost RGB-D sensors has led to an increased interest in object recognition combining both color and depth modalities. Object recognition from RGB-D images is particularly important in robotic tasks and the inclusion of depth has been proven to increase the performance. The problem of combining depth and color information is being widely researched. This thesis addresses this problem by initializing a 2-D Convolutional Neural Network (CNN) for RGB information via transfer learning and 3-D Convolutional Neural Network for encoding depth information. The obtained feature representations are fused to report performance over the RGB-D object recognition task. The transferred weights are from CNNs that are trained on large ImageNet classification challenge dataset and produces meaningful features. The depth information is encoded along with the color information in a 3-D voxel and learns joint features from scratch using a 3-D CNN. The approach is evaluated on the Washington RGB-D dataset and the performance for RGB category recognition exceeds the state-of-the-art, while the RGB-D performance is on par with it for category recognition. Due to good features learnt by the 3-D CNN, the potential of transfer learning from 2-D pre-trained CNN to 3-D CNN to include depth information is also addressed.

ÖZETÇE

RGB-D sensörlerinin son zamanlarda kolay ve ucuz temin edilebilir hale gelmesi, nesne tanıma alanında renk ve derinlik bilgilerinin ortak kullanımının yaygınlık kazanmasına yol açmıştır. RGB-D verisi ile nesne tanıma özellikle robotik alanı için oldukça önemlidir, ve derinlik bilgisinin kulanımının başarımı arttırdığı bilinmektedir. Renk ve derinlik bilgilerinin nesne tanıma için bir arada kullanılması yaygın bir araştırma konusudur. Bu tez çalışmasında bu problemin çözümüne yönelik olarak, renk bilgisi için önceden eğitilmiş iki boyutlu derin evrişimsel sinir ağları, derinlik bilgisi için ise üç boyutlu evrişimsel sinir ağları kullanılmıştır. Ayrı ayrı eğitilmiş iki boyutlu ve üç boyutlu sinir ağlarından elde edilen özniteliklerin tümleştirilmesiyle çeşitli nesne gösterimleri oluşturulmuş, ve bu gösterimlerin RGB-D nesne tanımadaki başarımları Washington RGB-D veri seti üzerinde sınanmıştır. ImageNet veri kümesi üzerinde önceden eğitilmiş derin evrişimsel sinir ağları aracılığıyla, RGB verisinden anlamlı gösterimler elde edilebilmiştir. Diğer yandan derinlik bilgisi renk bilgisiyle birlikte üç boyutlu bir voksel gösterimi olarak kodlanmış, ve üç boyutlu evrişimsel bir ağ yapısının bu gösterimi kullanarak sıfırdan eğitilmesiyle anlamlı ortak öznitelikler öğrenilmiştir. Washington RGB-D veri seti üzerinde elde edilen RGB kategori bazlı tanıma başarımının, literatürde şu ana dek ulaşılmış en iyi başarım sonucundan daha iyi olduğu görülmüştür. RGB-D bazlı tanıma başarımı ise literatürde elde edilmiş en iyi sonuca denk bir başarım sağlamıştır. Tez çalışmasında son olarak, önceden eğitilmiş iki boyutlu evrişimli sinir ağlarından üç boyutlu evrişimli ağlara bilgi aktarımı, potansiyel bir öğrenme transferi konusu olarak ele alınmıştır.

ACKNOWLEDGMENTS

I would like to express deepest gratitude to my advisor Assoc. Prof. Yücel Yemez for his full support, expert guidance, understanding and encouragement throughout my research. Without his incredible patience and counsel, this thesis would not have been possible. I also want to thank him for encouraging me to find a problem that I was interested in working on. His mentor ship has been inspirational. I also want to thank Assoc. Prof. Deniz Yüret for his constant help, support and encouragement throughout my thesis. Without his invaluable counsel it would have been really hard for me to advance in my research. I have gained incredible knowledge through his courses and advice, and I feel honored to have had the chance to learn from him. I also want to thank Prof. Bülent Sankur for agreeing to serve on my thesis committee. His feedback and comments are valuable for our research.

I am really thankful to my parents for everything that they have done for me. I am nothing without them. I want to thank my husband and best friend Hassan, who has supported me endlessly throughout this thesis. I am lucky that I get to spend the rest of my life with him.

Finally, I want to thank my friends Rizwan, Raza, Aatif, Tooba, Bushra, Sonia, Narjis, Naveed, Hüseyin, Buket and Shabbir for always being there, listening to my endless rantings and making my time in Istanbul memorable. I also want to thank the MVGL lab members for their help and support. Finally, I want to acknowledge the support of Koc University and TUBITAK which made the completion of this thesis possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Object recognition is an important problem in Computer Vision and is widely used in robotic tasks in physical environments. With easy availability of low-cost sensors like Microsoft Kinect, depth and color information can be simultaneously captured and included in recognition of objects in the physical environment. Depth provides additional information about the environment and has proven to improve the recognition performance when paired with color information. Unlike RGB images, depth images are invariant to light and allows better background separation.

Object recognition and classification has been extensively studied for RGB images, and there are large datasets available. Convolutional Neural Networks have been particularly successful and have produced state of the art results on these large datasets in challenges like ImageNet [Russakovsky et al., 2015]. The ImageNet challenge has led to development of successful deep convolutional neural networks for image classification like AlexNet, VGG net and GoogleNet [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014]. The availability of such models that produce meaningful features for RGB images are important for tasks that have smaller datasets available, since collection of large datasets for such tasks is time-consuming as well as require large amount of processing time while training. This paper focuses on utilizing pre-trained models and using their feature representations for robotic tasks involving small datasets that involve household objects.

While depth sensors are widely popular in robotics, there are no large scale dataset or models available as compared to those for color information. The topic of RGB-D object recognition is being widely researched on, but most of them focus on unsupervised learning and feature descriptors. We focus on the problem of incorporating depth information along with the RGB feature set to improve the performance.

We present an approach that exploits the RGB information learnt by large scale models particularly VGG net, encodes depth information in a spatial 3-D voxel for training a 3-D CNN from scratch and fuse the results to get the RGB-D performance. Our contributions include:

1. We exploit the information in the pre-trained VGG model and extract features representation from RGB images and train them with linear SVM for category and instance recognition. Our approach exceeds the state-of-the-art.

2. We study the problem of training depth images from scratch by training 3-D Convolutional Neural Networks. We pre-process the depth information to remove background and produce a 3D voxel from a depth image. We analyze the results incorporating RGB information in the 3-D voxels as well as fusing the features with RGB features from VGG net. Our fusion results are at par with the state-of-the-art for category recognition and exceeds it for instance recognition.

3. We study an approach that incorporated the depth information in the VGG network via transfer learning. We modify the first layer of the VGG network to input a 3-D voxel with pre-trained weights. We let the network train to learn new feature from the depth information.

This thesis is organized as follows: Chapter 2 discusses the related work, and Chapter 3 includes the methodology. Chapter 4 discusses the experimental setup.

We also analyze our results and provide a comparison with previous approaches and then present the conclusion in Chapter 5.

Chapter 2

RELATED WORK

There has been a great deal of work done on both RGB-D object recognition and convolutional neural networks. Recently, application of deep learning on RGB-D datasets has become very popular as well. This chapter covers a detailed discussion of previous approaches on RGB-D object recognition, convolutional neural networks and their joint application.

2.1 RGB-D Object Recognition

The previous methods suggested for RGB-D objected recognition are broadly divided into two categories: methods that use hand-designed descriptors and methods that learn features. These features are then fed into classifiers along with their labels for the final classification. The classifiers include SVMs and softmax regression.

2.1.1 Hand Designed Descriptors

Hand designed descriptors are widely used in computer vision tasks such as detection [Lai et al., 2011, Lowe, 1999], surface matching [Johnson and Hebert, 1999] and recognition [Lai et al., 2011, Lowe, 1999, Browatzki et al., 2011] One of the approaches include the use of SIFT descriptors [Lowe, 1999] and spin images
[Johnson and Hebert, 1999] for feature extraction, which are then trained using SVMs [Lai et al., 2011]. SIFT descriptors extract image features that are invariant to ro-

tation, translation and scale. These are thought to be similar to neurons found in inferior temporal cortex which are associated with human vision [Lowe, 1999]. On the other hand, spin images are computed on 3-D points with surface normals, resulting in a 2-D image (histogram) which encodes local information for each 3-D point on a surface. This information is then used for object recognition via correlation [Johnson and Hebert, 1999].

Other methods for color and depth images include depth kernel descriptors. These extract a set of kernel features that exploit size, 3-D shape information and depth edges simultaneously [Bo et al., 2011a]. SURF is another descriptor inspired from SIFT, but more robust than SIFT itself [Bay et al., 2008]. A number of descriptors are also histogram based. These include CSIFT, a variation of the SIFT descriptor that works for color images [Abdel-Hakim and Farag, 2006], Histogram of Oriented Gradients (HOG) descriptor that bins the orientation of gradients in locations of an image [Dalal and Triggs, 2005] and Signature of Histograms (SHOT) descriptor, which describes a point by computing histograms over a spherical volume superimposed by a three dimensional grid [Tombari et al., 2010].

Most of the above descriptors were originally designed for either images or 3-D points that exploited shape information. With the advent of color and depth sensors that can simultaneously record both modalities, these have been modified and variations of these descriptors have been developed to use both color and shape cues. A few examples include CSIFT [Abdel-Hakim and Farag, 2006] and 3-D SIFT [Scovanner et al., 2007] which are variations of SIFT descriptors modified for color and three dimensional information and CSHOT [Tombari et al., 2011] which a version of SHOT descriptor for both color and shape information. The problem with most hand crafted descriptors is that it is hard to extend them to apply to different modalities without modifications. They have to be tuned for specific datasets and also may not be able to fully use the available information.

A combination of these hand designed descriptors have been used to extract visual

and shape features including SIFT, Spin Images and HOG descriptors. The creators of the RGB-D dataset initially carried out evaluations using these descriptors, but since then feature learning methods have surpassed these results by a substantial increase in the performance [Lai et al., 2011].

2.1.2 Feature Learning

The second category involves feature learning from raw data. The learned features are again fed into classifiers such as SVMs for object recognition. Some of the methods that use feature learning include sparse coding, hierarchical matching pursuit [Bo et al., 2011b, Bo et al., 2013], convolutional k-means descriptors, transfer learning for convolutional neural networks, convolutional auto-encoders, regularized reconstructed ICA network and convolutional recursive neural networks. While there are numerous other works as well, we discuss those relevant to our work in detail.

One of the most fundamental methods devised for feature learning include hierarchical matching pursuit (HMP) algorithm. This work learns a dictionary consisting of code words. The idea behind this is to get a sparse representation of data as a linear combination of these code words. The dictionary is learnt using K-SVD in an iterative manner. Using this dictionary, the hierarchical matching pursuit builds a feature hierarchy recursively. The hierarchical matching pursuit algorithm consists of three main modules: batch orthogonal matching pursuit, pyramid max pooling and contrast normalization. This process is applied two times to extract features which are then fed into linear SVMs for classification [Bo et al., 2011b]. Originally this approach was introduced for gray-scale images, but is demonstrated to be applied to other modalities, specifically color and depth images. This is done by applying hierarchical matching pursuit algorithm on gray-scale, color, depth and surface normal information and these features are concatenated and fed into linear SVMs for RGB-D object recognition [Bo et al., 2013].

Another popular technique is called convolutional k-means descriptor. This involves steps like interest point detection and dictionary learning. First feature representations are learnt in an unsupervised manner. The main purpose is to learn these features on interest points hence patches are extracted around the interest points and k-means clustering is employed to learn a number of centroids building the dictionary. Feature responses are calculated over a neighborhood of the interest point and summed over four regions to constitute the final feature descriptor [Blum et al., 2012].

[Jhuo et al., 2014] proposed another method for feature learning which is known as Deep Regularized Independent Component Analysis (R2ICA). This is built on the idea of finding a meaningful relationship between the color and depth information. Hence, they propose to jointly learn weights for feature learning using gray-scale and depth images. This is following by spatial pooling and contrast normalization. The resulting features are then trained for classification using Support Vector Machines (SVMs).

We elaborate on other works involving convolutional neural networks in detail in the dedicated next section.

2.2 Convolutional Neural Networks

In recent years, deep learning has become extremely popular and has been extensively applied to computer vision tasks. Deep learning methods like convolutional neural networks [LeCun et al., 1998], deep belief networks [Hinton et al., 2006], and deep Boltzmann machines [Salakhutdinov and Hinton, 2009] have produced good results when applied to visual analysis. Among these methods, convolutional neural networks are being popularly used to solve vision related task such as scene labeling [Farabet et al., 2013], object recognition [socher], face verification [Taigman et al., 2014] and pose estimation [Toshev and Szegedy, 2014]. Convolutional neural networks were originally introduced by LeCun [LeCun et al., 1998] for a hand written digit recog-

nition problem. Since then they have been used on dataset that include rich images like ImageNet and have achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge [Russakovsky et al., 2015]. Recently, application of convolutional neural networks for data has become popular and various methods have been suggested to achieve superior performance as compared to hand-designed descriptors.

2.2.1 Convolutional Neural Networks for RGB-D Object Recognition

Convolutional neural networks have been used in combination with other modalities to solve the RGB-D object recognition. One such technique is a combination of convolutional and recursive neural networks that are based on the idea that convolutional layers extract low level features and recursive neural networks extract the high level features. The method consists of pre-processing that includes local normalization and whitening, convolution layer, following by local contrast normalization, pooling and a tree of recursive neural networks. The convolution layers weights are pre-trained using the k-means clustering algorithm and this is performed for depth and color images separately. The result is combined before feeding in a softmax classifier for recognition [Socher et al., 2012].

Another work adapts the above mentioned approach and modifies it to achieve better performance. This is done by proposing a semi-supervised framework using co training, which uses less labeled data but achieves competitive results as compared to the state of the art. Also, the architecture is modified by adding spatial pyramid pooling (SPM) to the convolutional recursive neural network model [Bo et al., 2013]. This is to solve the problem of variable dimensions of the dataset instead of re-sizing it to the same size [Cheng et al., 2015].

The above mentioned methods train depth and color modalities separately and combine them for performance of the RGB-D data. This may not be the correct way

to exploit the correlation between the two modes. Hence, some works focus on joint learning to explore this relationship and if it benefits the accuracy. One technique proposes a deep regularized reconstruction Independent Component Analysis (ICA) network. This jointly learns filters from unlabeled gray-scale and depth data and applies these learnt weights to both of them. This is followed by spatial maximum pooling and local contrast normalization. This, repeated for three layers followed by an SVM classifier is the model [Jhuo et al., 2014] for this method which produces one of the best results on the RGB-D dataset [Lai et al., 2011].

A recent work tackling the same problem of joint learning introduces a multi-model layer to a CNN-based neural network. This is done by constructing separate networks for color and depth data using convolution and pooling layers. These networks are then joined together at the fully connected layers followed by a multi-model layer that learns features from both networks iteratively. Back-propagation is performed on the fully connected layers. This idea is built on exploring the discriminative features as well as correlated features of depth and color information and performs well on the RGB-D dataset [Lai et al., 2011, Wang et al., 2015].

Another interesting work involves the use of a convolutional neural network (AlexNet) [Krizhevsky et al., 2012] pre-trained on color images. The same network is also use for depth images by converting them in color images. This is done by rendering objects canonically by colorizing them according to their distance from the center Features from the last two layers of this model are them combined and fed into an SVM to yield superior results [Schwarz et al., 2015]. Another method that improves this approach modifies the depth image to an RGB images using color map and introduced multi-modal learning to learn joint features from the pre-trained AlexNet. This approach outperforms in the RGB-D category recognition [Eitel et al., 2015].

Transfer learning is also an idea that adds to the list of exploring relationship between depth and color information. This technique involves transfer learning of convolutional neural networks across channels (red, green, blue and depth). Instead

of training networks in parallel, which is mostly done in the works discussed, this approach sequentially trains networks. This is done by using the trained weights of a network as a starting point for the training of the second network [Alexandre, 2016]

Using pre-trained weights for initializing a deep network has proven to be useful especially when the dataset is small and cannot learn a deep network from scratch. [Yosinski et al., 2014] explores the benefits of transfer learning by initializing partial networks from trained deep networks and shows that initializing weights from trained networks are always helpful than initializing them randomly. This is true even if the target dataset and task is different from those of the trained network. This is because the initial layer of a deep network learn features that can be generalized. They also study the effect of freezing the initialized layers and not updating them during training to show that fine-tuning and updating these layers result in better performance.

2.2.2 3D Convolutional Neural Networks

Due to the availability of faster GPUs and dedicated Cuda libraries for deep learning (cuDNN), 3-D dimensional convolutional neural networks are now increasingly becoming popular. 3-D CNNs are being increasingly used for region proposal, object recognition and medical imaging. 3-D CNNs allow combined representation of depth and RGB information, with depth representing the extra dimension.

One such work focuses on neuroimaging using 3-D MRI scans to predict Alzheimer's disease. The network consist of 3-D convolutional layer pre-trained via unsupervised learning using sparse auto-encoders. They also implement the same network for 2-D data and conclude that 3-D representation boosts the performance when compared [Payan and Montana, 2015].

For medical images 3-D scans are available for input to a 3-D network, but for

robotic tasks such data is only available in the form of RGB and depth images. These images can be combined to generate point clouds, which can be rendered to produce object surfaces or their features can be separately learnt and then combined for RGB-D objection recognition performance. 3-D networks provide the opportunity to combine the input information and then learn features, and the problem of combining depth and RGB information in a 3-D form is a popular one.

A couple of approaches have been suggested to get the 3-D grid/voxel form from depth and RGB images to be fed into 3-D networks. One such approach, called ShapeNets represents the depth information into a grid in the form of truncated signed distance function (TSDF). Each value in a voxel represents the shortest distance between the voxel and the nearest surface point. This is computed in three directions and is called the directional TSDF. Combined with RGB information this yields significant performance in region proposal and object recognition tasks [Song and Xiao, 2015].

VoxNets is another approach that encodes RGB and depth information into a volumetric occupancy grid. They propose three methods of forming the grid that include binary occupancy grid where each voxel has binary state either occupied or unoccupied, and density occupancy grid which is a probabilistic method for formulating the grid. The last method is hit grid, which a simple method to record hits for each voxel, while not distinguishing between unoccupied and missing voxel information. They report that this approach produces surprisingly good results although it discards a lot of information [Maturana and Scherer, 2015].

Chapter 3

METHODOLOGY

3.1 RGB Feature Extraction Pipeline

The existence of competitions like ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Russakovsky et al., 2015] has led to extensive research when it comes to object classification/recognition using RGB images. For smaller datasets, training a deep network from scratch does not produce comparable results. Hence, either unsupervised learning is used to learn weights for initial layers or weights from large trained networks are needed for good performance. The latter also saves the training time for large scale deep networks. We have worked on both the approaches.

3.1.1 Unsupervised Learning for Initializing CNNs

Unsupervised Learning is a popular approach used for training the initial layers of deep networks. This provides a good initialization for training the rest of the layers and improves the performance as compared to random initialization. Our unsupervised approach is based on that of [Socher et al., 2012], although we work on a much smaller resolution of 50x50. Figure 3.1 shows the architecture of our approach. The first layer is the convolution layer that convolves K filters to extract features into K feature maps. The filters used for convolution are learnt by running k-means clustering on randomly extracted patches from the dataset. This learns K centroids from the data, that are standard edge and color features [Coates et al., 2011].

This is then followed by Rectified Linear Units(ReLUs) layer that convert all negative values to zero[Nair and Hinton, 2010]. Local Contrast Normalization(LCN) is then performed which is a subtractive and divisive normalization that contrasts features within feature maps as well as across feature maps in the same spatial locality [Jarrett et al., 2009]. A max pooling layer is then applied that performs sub-sampling to reduce dimension of the response. Instead of using randomly initialized Recurrent Neural Networks following the pooling layer as done in [Socher et al., 2012], we add two randomly initialized fully connected layers. We also include dropout layers before and after the first fully connected layer to reduce over fitting. Dropout layer drops each element of the response with a probability p, and replaces it with a zero value [Srivastava et al., 2014].All the images are re-scaled to 50x50 and go through local mean subtraction and deviation normalization and ZCA based whitening to remove correlation [Hyvärinen and Oja, 2000].

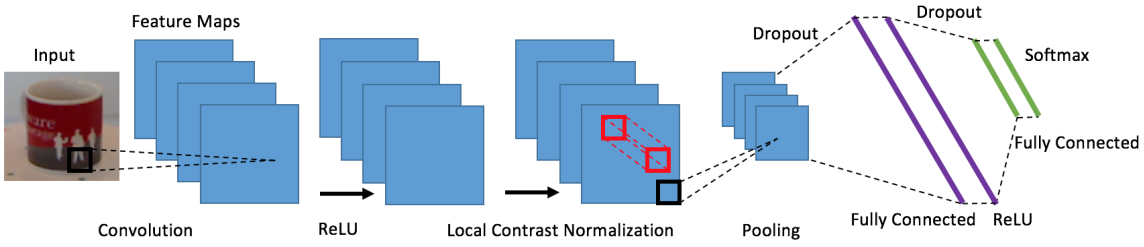

Figure 3.1: RGB Unsupervised Learning Architecture

We evaluate the performance of the network layer by layer, with and without pre-training. We evaluate the performance using only convolution layer followed by rectification and pooling using the filters learnt from k-means clustering. The same process is repeated with filters that are randomly initialized using Xavier initialization. Xavier initialization picks random weights from a uniform distribution whose range is defined by the output size of the previous layer [Glorot and Bengio, 2010]. We compare results from these experiments to evaluate how much pre-trained weights add to our performance. We then add the local contrast normalization to this network

configuration to study its effect on the performance as well. Finally, we run the fully architecture to observe how the performance changes when a randomly initialized layer is added on top of the pre-trained layer.

3.1.2 Pre-trained Deep Networks

We investigate the VGGnet [Simonyan and Zisserman, 2014], the runner-up ILSVRC 2014 model. The VGGnet is a very deep network that consists of stacks of convolution layers each with very small filters (3x3) that has beaten the previous state-of-the-art on the ImageNet [Russakovsky et al., 2015] classification and localization challenge. Unlike previous proposed models, the VGG net reduces the training parameters by using smaller filter sizes combined with a larger depth to achieve these results.

The 16-layer trained VGG net is used for all the experiments. The structure of the VGG network is shown in Figure 3.2. The architecture consists of stacks of convolution layers followed by maxpooling layer. The number of feature maps increase as the depth increases and the convolution layers are followed by three fully connected layers. In Figure 3.2, for an N x N x n input (n is the number of input channels), ConvA-B means convolving the 2-D input with B filters of size A x A each (stride, s and padding, pd is set to 1 for each layer), resulting in a response of dimension $i = \frac{N-A+1+pd}{s}$ and size i x i x B. This is followed by adding bias of size 1 x 1 x B and applying ReLU rectification. Pool-C means max pooling the response in square region of size C x C resulting in an output of dimension $j = \frac{N-C}{S} + 1$ and size j x j x n. FC-D is a fully connected layer that outputs a response of size D after addition of bias of size D and ReLU rectification. The softmax layer is the classifier that is used for training the VGG net.

While the initial layers learn general features, the last layers learn more dataset specific features. To extract the most meaningful features for our task, we investigate the last three fully connected layers of the VGG net to understand where the networks

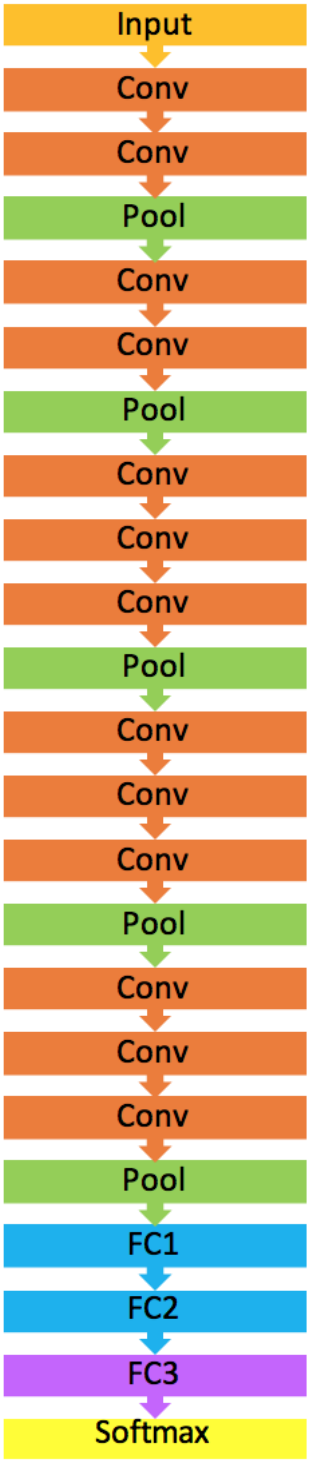

Figure 3.2: Architecture of the 16-layer VGG net

inclines towards learning more dataset specific features. This is to ensure we make best use of the VGG net for our task. For this, we remove the last n layers, where $n = 1..3$ from the VGG net and extract features without training the rest of the network. The dataset is convolved/multiplied with the layers of the partial network without backpropagation or learning and the resulting features are used for training linear SVM for each of the experiment. Because [Schwarz et al., 2015] suggests fusion of features obtained from the last two layers, we also evaluate the performance by combining the features extracted from each of the network configuration described and comparing their performance.

The dataset is re-scaled to the input size of the net(224x224). Following the VGGnet pre-processing, the mean training pixel values for each channel are subtracted from the dataset. Besides this, no other pre-processing is required.

As the final architecture for the RGB recognition task, VGGnet upto the first fully connected layer is used as shown in Figure 3.3. After extracting the features from the proposed network (no fine-tuning), linear Support Vector Machines (SVM) are used as the classifier.

3.2 3-D Feature Extraction Pipeline

While 2-D CNNs are a natural way to encode RGB information, the question that arises is to encode depth information. While previous approaches make use of pre-trained RGB networks by representing depth maps as color images [Schwarz et al., 2015], we propose to encode depth information in the most natural way as possible preserving the spatial information. Rather than encoding the depth information as any function or descriptor [Song and Xiao, 2015], we propose to represent the depth information as raw as possible and investigate if a CNN can learn meaningful representations.

We encode the depth information in a 3-D voxel by building a third dimension

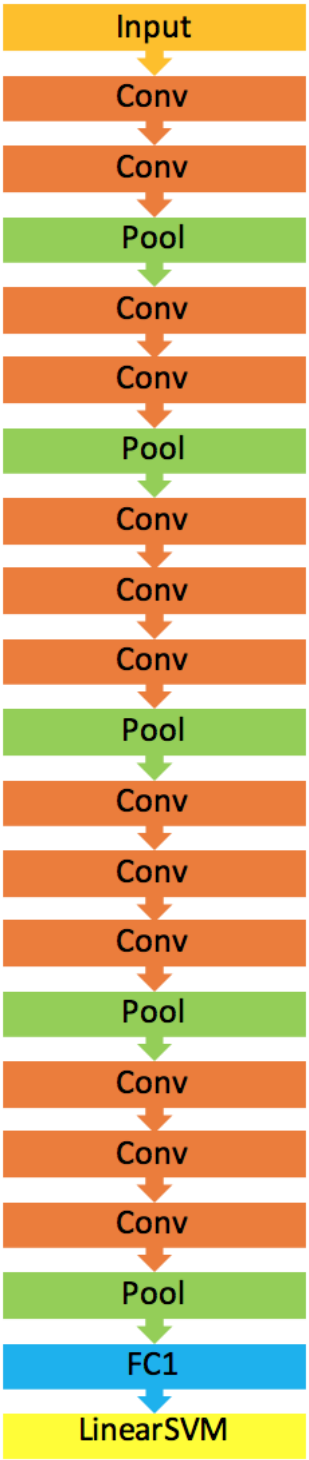

Figure 3.3: Proposed architecture for RGB recognition task

based on depth values present in a depth image. Each pixel at depth image stores the corresponding depth value for that position, while the equivalent color image stores its RGB values. We create a 3-D voxel, with similar length and width as the original image, and height as the difference of the maximum and minimum depth value found in the image. Each RGB pixel is then placed in the same position in 3-D voxel but at its corresponding height. This results in a 3-D voxel that encodes the spatial information between each pixel. Encoding the RGB information helps to jointly learn features that are related to both depth and color rather than learning features from depth only.

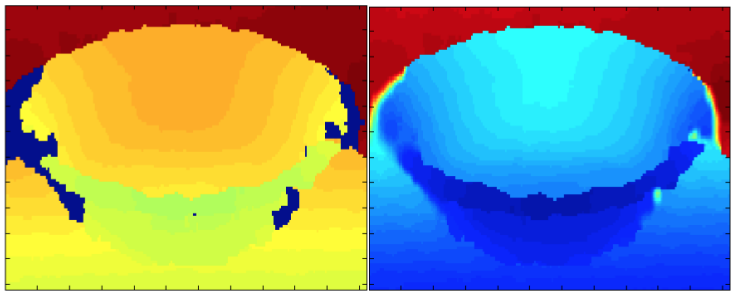

Figure 3.4: (Left) Original image, with blue color denoting missing values. (Right) Interpolated Image with filled approximated values.

The depth images in the dataset have missing values in some regions where the depth sensor is not able to capture properly. We process these images by doing an interpolation to fill the missing values. Figure 3.4 shows the result of the interpolation. We then apply the provided segmentation mask to filter out the background information and only to encode object shape since the turntable in the background has similar depth values to the object and interfere with its shape. The result is then converted to a 3-D voxel and re-scaled to size 30x30x30. We use a smaller dimension due to computation and memory constraints while training a 3-D CNN.

We follow the VGGnet [Simonyan and Zisserman, 2014] in terms of architecture while designing the 3-D CNN. The filter size are kept small and dimensions are main-

tained after each convolution layer to retain information since the dimensions are already smaller as compared to the original object sizes. Two convolutional layers are added with each layer following a maxpooling layer and non-linearity. Figure 3.5 shows the final network. The weights are randomly initialized using Xavier initialization and the network is trained using backpropagation.

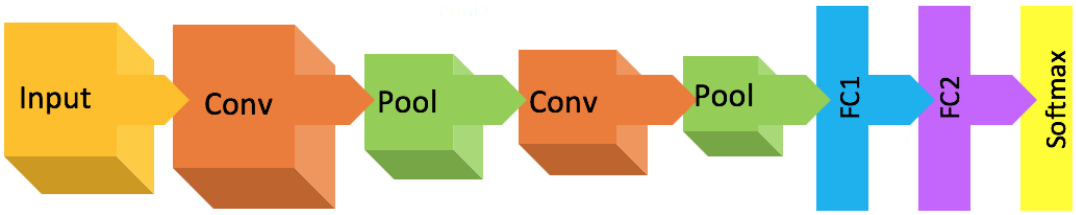

Figure 3.5: Proposed 3-D Network

3.3 RGB and 3-D Fusion

We fuse the outputs of the pre-trained VGGnet and our 3D CNN and train them to get our final RGB-D performance. We do this because this ensures that we include the meaningful RGB representations from the pre-trained VGGnet along with the 3-D CNN. Although, the 3-D CNN contains RGB information, it is trained on a much lower resolution than the VGGnet resulting in a loss of RGB information. Also, the 3-D CNN is trained from random initialization. Hence, we fuse the results of both the networks and train it with linear SVM and report the performance. Figure 3.6 shows the final network configuration.

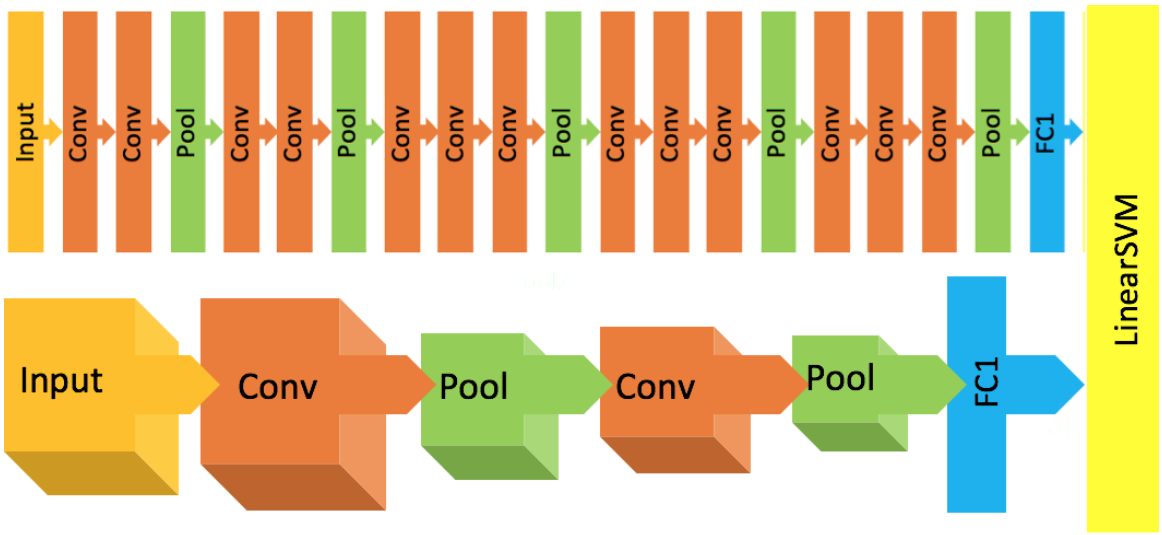

Figure 3.6: Proposed fusion framework for RGB-D object recognition

3.4 Transfer learning from 2-D to 3-D

Transfer learning involves reusing learnt weights from a trained network to initialize another network that trains a dataset for another task. It is especially helpful if the datasets are small and deep networks can not be trained from scratch without over fitting. It has also proven to be useful where tasks and dataset are not same since it provides a good initialization for initial layers that learn general features rather than task specific features [Yosinski et al., 2014]. We are also able to achieve better performance using transfer learning from our RGB pre-trained network (VGGnet) as compared to training a deep network from scratch. Inspired from this, we study the problem of transfer learning from 2-D to 3-D to incorporate depth information in the already well-performing RGB pre-trained network. While transfer learning has been applied on depth data by converting it into an RGB image, initializing a 3-D network from a pre-trained 2-D network is an open problem.

To do this, we propose to modify the VGGnet for a 3-D voxel input that encodes depth and color information. We call this modified VGGnet as VGG3-D. Since the de-

fault input to VGGnet is 224x224x3, due to memory and computational constraints, we encode depth in a non-uniform distribution that spans the third dimension, resulting in a 224x224xdx3 voxel input. Here d represents the newly added depth dimension. We choose d=6 for our experiments.

The depth values over the whole dataset are analyzed to find d-1 intervals, such that each interval has equal number of depth points. The depth values are binned using the found intervals, which are then quantized in the depth dimension d. While the intervals contain equal number of depth points over the whole dataset, individual instances have different number of depth points present in each interval. Although we retain spatial information, we lose specific depth information within each depth interval. This is a trade off to computational and memory cost. The last depth slice d in the resulting voxel contains the background depth values that we compute using the inverse of the segmentation mask provided. Figure 3.7 shows the result of this pre-processing, it displays a 3-D slice from the voxel to show the inclusion of depth information in an image. Figure 3.8 shows the proposed 3-D transfer learning framework.

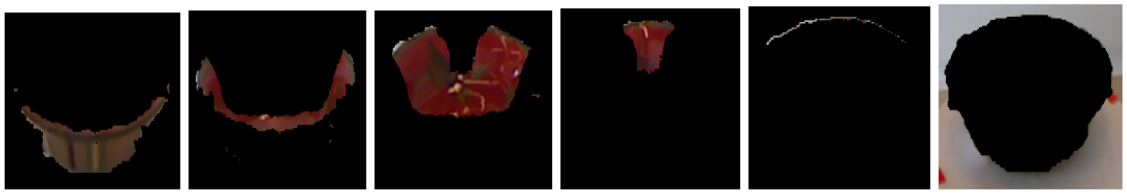

Figure 3.7: The resulting 3D voxel from incorporating depth into RGB image shown as individual 3-D slices

The purpose of transfer learning is to provide good initialization for training a deep network. To ensure we can make use of the initialization from the 2-D VGGnet for VGG-3D, we replicate the filters of the first layer of VGG-3D in the depth dimension d. We do this so that when we convolve our 3-D voxel input with the replicated weights, we are able to get the same response as we get after the first layer from the

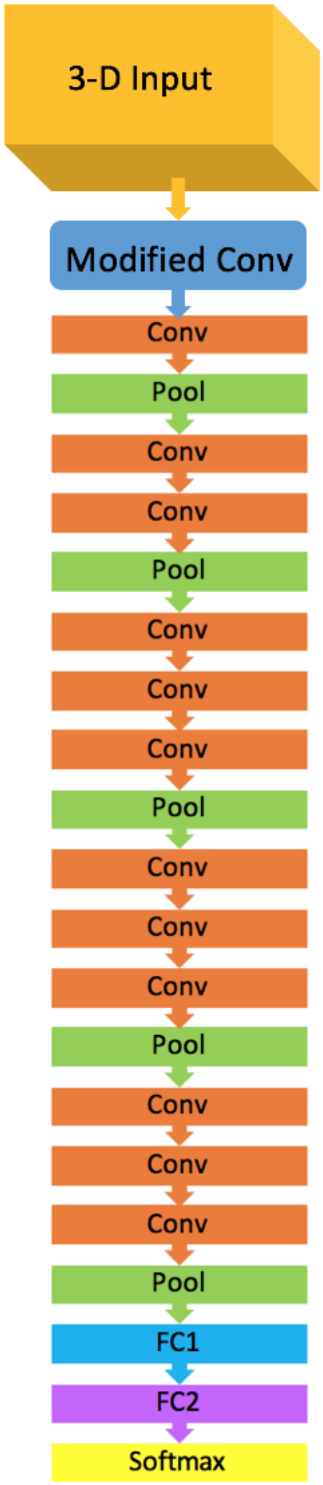

Figure 3.8: Proposed 3-D Transfer Learning Framework

2-D VGGnet. The response is consistent with the rest of the network and hence gives us a starting performance as good as the 2-D VGGnet.

The voxelization process introduces another dimension to the input. Since VGGnet is for RGB images, it expects a 2-D input and has 2-D weights for each layer. Figure 3.9 (i) shows a 2-D input with one channel and a binary 2-D filter and its convolved response. Because a 3-D voxel is going to be the input in VGG3-D, the weights or filters of the first convolution layer also need to be replicated along the depth dimension d to replicate the same response as in the original 2-D VGGnet. Figure 3.9 (ii) shows a sample 3-D voxel constructed by including depth information in the input from Figure 3.8 (i). It also shows the 2-D binary filter from Figure 3.9 (i) that is replicated in the depth dimension d for a 3-D convolution. This is based on the argument that for every (x, y) pixel in an image, there will only be one non-empty voxel value along the dimension d. Hence when weights are replicated along d to form 3-D filters, the output of the 3-D convolution will be exactly the same for the first layer as it is in 2-D. From there on, each of the replicated depth map weights are independently modified by back propagation to learn more information from the depth. The resulting response is also shown, which is exactly the same as that with the original 2-D input and filter.

Since we do not need to modify the rest of the VGGnet, the first layer is a 3-D layer while the rest of the net stays 2-D. Because 3-D convolution is more expensive than 2-D and also its response has to be reshaped to be compatible with the rest of the network, we come up with a method to represent the 3-D voxel in 2-D form.

This is done by representing each of the 3-D slice in the voxel as a separate channel or feature map. Hence, the original VGGnet has a 3-channel input (for R,G,and B) each of which is individually converted to a 3-D voxel in VGG3-D. When representing the 3-D voxel in 2-D form, each voxel slice becomes a separate channel resulting in a 3 times d channel input. For $d = 6$, this becomes a 18-channel input in VGG-3D. For simplicity, in Figure 3.9 a 1-channel input is used. Figure 3.9 (iii) shows each

3-D slice in the voxel in (ii) being represented as six separate channels. Likewise, the 3-D filter in (ii) is also represented as six 2-D filters. Since all channels are convolved and summed, when these six channels are convolved with the six filters, their sum produce identical response to the 3-D convolution. We use this method to represent our 3-D information in VGG3-D. Since we start from a good initialization, we try to fine-tune the network with 3D input to learn new depth-based features on top of the already learnt RGB features.

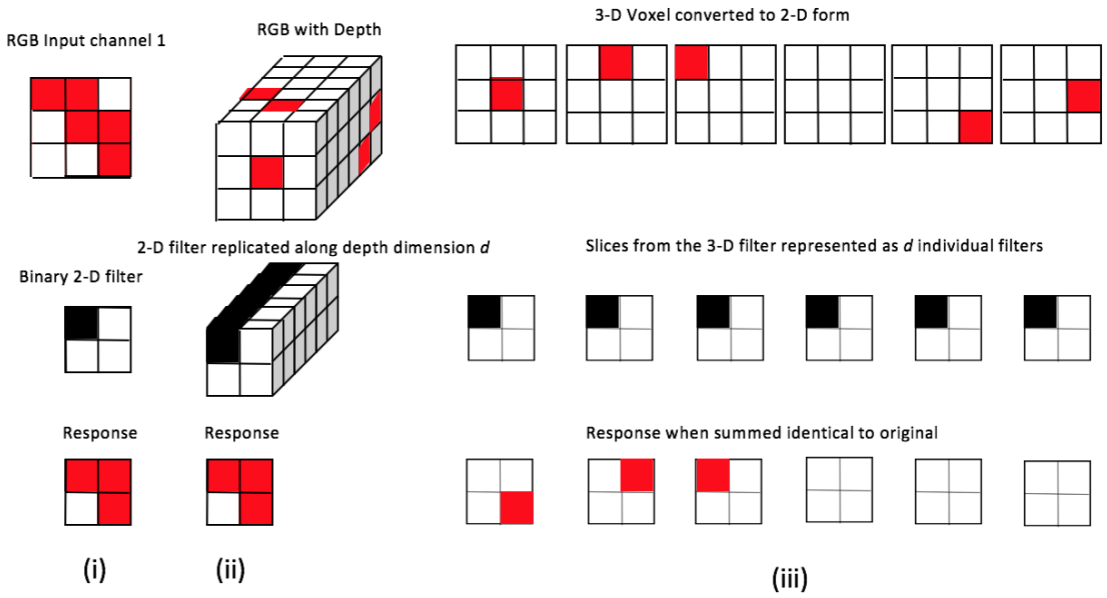

Figure 3.9: (i) 1-channel 2-D input convolved with a binary 2-D filter and its response. (ii) 2-D input converted to 3-D voxel by including depth, with 2-D filter replicated along depth dimension d to get a 3-D filter and their convolution response. (iii) 3-D voxel slices converted to individual channels, 3-D filter also converted to d individual filters and their individual responses. When summed, the response is identical to that of 3-D voxel and filter.

Chapter 4

EXPERIMENTATION

4.1 The Dataset

All experiments are conducted on the RGB-D Dataset by [Lai et al., 2011]. This is a large dataset that is captured using a Kinect style camera, that records RGB and depth video sequences simultaneously. It consists of 300 small household objects like fruits, vegetables, boxes and water bottle. Each of these objects are instances of 51 categories. Hence, the dataset is group by category as well as instance. For examples, apple is a category that has five instances each of which can be red, green or yellow.

Figure 4.1: Example of some objects segmented from the background

Each instance has depth and RGB images from three video sequences. One video sequence consists of a full turn table rotation with placing the camera at a certain angle, while the object is kept stationary. The video sequences are filmed with camera at 30°, 45°and 60°. The cropped version of the dataset is used which consists of bounding box images, along with the segmentation mask which filters the background from both the depth and RGB image. The evaluation dataset consists of every fifth frame of each of the video sequence resulting in roughly 42000 images. Out of these around 35000 instances are used for training and 7000 for testing.

The experiments on the dataset are carried out for category recognition and instance recognition. We focus on both category and instance recognition. The category recognition experiments are carried on ten splits. Each split has randomly select object from each category in the test set and the remaining 300-51=249 objects in the training set. The performance is reported over all ten splits as followed by [Lai et al., 2011].

Figure 4.2: Instances from the category 'Ball'

This dataset is tricky because it has unique instances for each object. For example, the object ball can have instances that are not only of different colors but also of different shapes and size. The instances include tennis ball, golf ball and baseball. For category recognition experiments, the testing set includes instances that are completely different from the training set. Also, some round objects like tomato and sponge that are similar in both shape and color are also harder to tell apart.

In instance recognition, training is carried in a leave-sequence-out manner, where images from the video sequence where camera is mounted at 30°and 60°are part of the training set and the video sequence for 45°becomes the testing set. Here, some instances that are very similar with a category, like fruits and vegetables are hard to tell apart, while objects like soda cans that have unique labeling are easy to separate.

4.2 Experimental Setup

For all category recognition experiments, validation and parameter optimization is done on split 1 and is repeated with fixed settings over the rest of the splits. The instance recognition experiments are also carried out on the same network and parameter setting as the category recognition. For RGB unsupervised learning experiments, we optimize the convolution layer's filter size and number of feature maps, pooling size and learning rate for one layer experiment using hyperopt [Bergstra et al., 2013]. The optimized settings for convolution window size and pooling size are fixed for pre-training, LCN and two layer experiments. In two layer experiments the size of the fully connected layer, dropout probabilities and learning rates are again optimized.

For pre-trained deep learning for RGB images, the features from the VGGnet are extracted without fine tuning and fed into linearSVM. The cost parameter for the linearSVM is optimized on the development set for Split 1 of the category recognition task and is fixed for the rest of the folds and instance recognition task. The same protocol is followed for the fusion framework. The feature size is 4096.

For 3-D CNN training, we fixed the learning rate to 0.1 and number of epochs to 10 for all the ten folds. The network is trained using back-propagation. The trained network is used to extract features which are optimized using linear SVM to get the performance. The linear SVM optimization is the same as for pre-trained RGB task. The extracted features of size 4096 are then used for the fusion framework as well. The fusion experiments have a feature size of 4096 + 4096 = 8192.

For transfer learning problem, the same configuration of VGGnet is used. The last two layers are eliminated, and a softmax layer is added for back-propagation. Instead of randomly initializing the softmax layer, the weights of the VGGnet are fixed and the softmax layer is trained on the development set. This learnt layer is then used with VGGnet for fine-tuning. For fine-tuning, we tried varying learning rates and dropout. The network's learning rate is set a much smaller value of 0.00001. However, since the first layer is modified, its learning rate is set 10 times higher than that of the rest of the network. The fine-tuned network is then used for feature extraction to get new features. These features are trained with linear SVM and also fused with the RGB pre-trained network to compare the performance.

4.3 Results

4.3.1 Unsupervised RGB Learning

Table 5.1 shows the results for RGB unsupervised learning. All the results were computed with back-propagation through softmax layer only. There is a general trend of performance drop if the error is back-propagated through the whole network. Pre-training increases the performance by almost 4% and LCN further adds another 4% to the performance. Adding a fully connected layer without the LCN layer does not result in a significant increase, but boost the performance if the LCN layer is included by more than 5%. Adding another layer to this network resulted in a drop in the performance. Even with randomly initialized fully connected layer and no back-propagation, we are able to get an increase in performance as compared to one layer due to good initialization of the first layer. With this network we are able to get as good performance as [Socher et al., 2012]'s proposed CNN-RNN based approach on split 1.

Experiment	RGB Performance
1 conv layer	69.8
Pre-trained conv layer	74.0
Pre-trained conv layer + LCN	77.8
Pre-trained conv layer + FC + Dropout	74.4
Pre-trained conv layer + LCN + FC + Dropout	79.9

Table 4.1: Results of 2-D RGB framework over Split 1, FC is a Fully connected layer

4.3.2 RGB pre-trained network, 3-D CNN and Fusion

Table 4.2 shows a comparison between feature extraction from different levels of the VGGnet. Firstly, we observe that there is a huge boost in performance from the unsupervised learning results (more than 10%). This is because all the layers of the VGGnet are trained on a very large and diverse dataset and it has learnt meaningful features that are applicable to other datasets. Secondly, when we compare the performance of features from different levels of the VGGnet, its observed that as we move closer to the last level of the VGGnet, the performance worsens. This is because the most lower layers of the network learn features specific to the object categories of the original dataset. This first fully connected layer performs the best as anticipated. Performance in the second fully connected layer starts to worsen and in the last fully connected layer it drops below our worst unsupervised learning performance.

We also do a comparison while to fuse the features of the fully connected layers and train them as done by [Schwarz et al., 2015]. We observe that when the best performing layers are fused we get an increase of 0.03% in the performance. Fusing the other combinations do not result in any increase. Since the feature size doubles while fusing, we decide to only use the best performing layer as there is not very signification increase when the features from the second fully connected layer are

fused. We use this network to compute our RGB results.

VGG Layer	Feature Size	RGB Performance
FC1	4096	91.08
FC2	4096	89.25
FC3	1000	72.24
FC1+FC2	8192	91.11
FC2+FC3	5096	89.20

Table 4.2: A comparison of performance using features from fully connected VGG layers over Split1, where FC-n in nth fully connected layer

Table 4.3 shows the ten-fold results for RGB, 3-D and fusion frameworks. It reports the mean accuracy and standard deviation as well. The RGB framework results show good performance of around 89% while the 3-D framework performs around 78%. However, the 3-D features add give a significant 2.5% boost to the RGB results when fused together and trained.

There are a few reasons why 3-D, although having both depth and RGB information does not perform better than RGB itself. Firstly, unlike RGB it is trained from randomly initialized features and thus can not take advantage of prior learnt information. Secondly, 3-D has a very small dataset size as compared to RGB input. Due to this, there is a loss of both RGB and depth information while re-scaling the dataset to a very small size. But when fused with RGB, the 3-D features that jointly learnt information from depth and RGB both gets meaningful features from the RGB pre-trained network and thus add significantly to the RGB performance.

The same frameworks are also used for instance recognition task. This is different from the category recognition task since it classifies individual objects rather than categories. Table 4.4 shows the results for RGB, 3-D and fusion for instance recognition. The performance for RGB is better for instance recognition is better than category

Split	RGB	3-D	Fusion
1	91.04	76.33	91.03
2	92.69	76.88	92.09
3	86.15	79.96	90.90
4	87.62	74.69	90.27
5	88.84	78.63	92.39
6	89.72	79.61	90.40
7	90.70	83.12	92.57
8	87.87	77.40	91.64
9	88.79	77.40	90.35
10	86.15	80.30	91.56
Mean	**88.96**	**78.43**	**91.29**
Dev	2.13	2.41	0.86

Table 4.3: Results of individual and mean ten-fold split validation for RGB,3-D and Fusion pipeline for category recognition

recognition but worse for the 3-D framework. Since most household category objects were different from each other in color, they seem to be easily recognized. However, instances in the same category are very similar in shape and color, hence they can be easily confused. For example, vegetables and fruits like tomato, orange have the same color and shape for all instances. Hence, the 3-D framework performance is low, and when fused together the overall performance drops by 0.4% when compared to RGB results.

RGB	3-D	Fusion
95.01	26.74	94.60

Table 4.4: Results for RGB,3-D and Fusion for instance recognition

4.3.3 Transfer Learning Experiments

We fine-tune our modified VGGnet (VGG3-D) in an attempt to learn new features from depth information that is incorporated. Table 4.5 show the results for split 1. We compare the performance with our 3-D CNN trained from scratch and also their fusion result with RGB pre-trained network. As the results show, the VGG3-D performs worse than the original RGB pre-trained network, but better than the 3-D CNN. However, surprisingly, when fused with RGB pre-trained network framework features, the 3-D CNN performs better than when VGG3-D is fused with the same RGB pre-trained network features. This can be due to the fact that 3-D CNN learns more depth features which are different from the RGB features and hence add on top of the performance, while VGG3-D is not able to do that.

Fine-tuning a deep network is tricky and over fitting becomes a problem once the network weights have converged in one direction, hence it is possible that the VGG3-D does not learn any new features. Also, they depth dimension is much smaller as compared to 3-D CNN, hence we are losing more information regarding depth in VGG3-D.

Method	Performance
RGB pre-trained network (original framework)	91.04
Transfer Learning (VGG3-D)	88.70
3-D CNN	76.33
Fusion (RGB + VGG3-D)	90.70
Fusion (RGB + 3-D CNN)	91.03

Table 4.5: A comparison of performance with split 1 on category recognition between our VGG3-D, RGB and 3-D CNN framework

4.3.4 Comparison with previous methods

Table 4.6 and 4.7 compares our approach with previous methods for the category recognition and instance recognition task. Our approach exceeds the previous performance in all tasks except for RGB-D object recognition where it is at par with the method of [Eitel et al., 2015].

Method	RGB	Extra Information	RGB-D
[Lai et al., 2011]	74.30 ± 3.3	53.1 ± 1.7 (D)	81.90 ± 2.8
[Bo et al., 2013]	82.40 ± 3.1	81.2 ± 2.3 (D)	87.50 ± 2.9
[Schwarz et al., 2015]	83.10 ± 2.0	N/A	89.40 ± 1.3
[Socher et al., 2012]	80.80 ±	78.90 ± 3.8 (D)	86.80 ± 3.3
[Jhuo et al., 2014]	85.65 ± 2.7	**83.94 ± 2.8** (D)	89.59 ± 3.8
[Cheng et al., 2015]	85.20 ±1.2	81.2 ± 2.3 (D)	90.10 ± 1.1
[Eitel et al., 2015]	84.10 ± 2.7	83.8 ± 2.7 (D)	**91.30± 1.4**
[Wang et al., 2015]	74.6 ± 2.7	N/A	86.90 ± 2.9
Our RGB+3-D	**88.96 ± 2.1**	78.43 ± 2.4 (3-D)	91.29 ± 0.86

Table 4.6: Comparison of performance with previous methods for category recognition

Method	RGB	Extra Features	RGB-D
[Lai et al., 2011]	59.3	32.3 (D)	73.9
[Bo et al., 2013]	92.1	51.7 (D)	92.8
[Schwarz et al., 2015]	92.0	N/A	94.1
[Jhuo et al., 2014]	92.43	**55.69** (D)	93.23
Ours RGB+3-D	**95.0**	26.7 (3-D)	**94.8**

Table 4.7: Comparison of performance with previous methods for instance recognition

Chapter 5

CONCLUSION

We study the problem of learning information from both RGB and Depth for object recognition. Several methods were tried. While unsupervised learning for initializing the first layer for CNNs helps boost the performance, along with Local Contrast Normalization (LCN) and dropout, pre-trained deep neural networks like VGGnet perform significantly better. These networks produce meaningful representations for RGB images and gives state-of-the-art performance on the RGB recognition tasks.

Since our dataset is much smaller than one used for training the VGGnet, we were able to use it as a feature extractor rather fine-tuning it which might lead to over fitting. For larger datasets, fine-tuning the pre-trained network is an option to boost performance. For datasets that are smaller and very different, unsupervised learning and LCN can be helpful in training smaller networks.

Incorporating depth information along with RGB is tricky. We try to encode depth information in a 3-D voxel preserving geometric and spatial information. We incorporate RGB information in the voxel to jointly learn features using a 3-D CNN on a much smaller resolution than the RGB pre-trained network. This learns good representations, which when combined with the RGB pre-trained network features are valuable in increasing the performance to cross 90% and is comparable with the state-of-the-art.

We believe that pre-trained networks contain valuable information and 3-D CNN can learn good depth cues. Hence, we experiment on a modified VGGnet, VGG3-D to

combine both of these networks in a more meaningful way. At this point, the results do not surpass our RGB+3-D CNN fusion framework. However, as part of the future work, we plan to experiment with different training routines to improve the VGG3-D performance.

BIBLIOGRAPHY

[Abdel-Hakim and Farag, 2006] Abdel-Hakim, A. E. and Farag, A. A. (2006). Csift: A sift descriptor with color invariant characteristics. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1978–1983. IEEE.

[Alexandre, 2016] Alexandre, L. A. (2016). 3d object recognition using convolutional neural networks with transfer learning between input channels. In *Intelligent Autonomous Systems 13*, pages 889–898. Springer.

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359.

[Bergstra et al., 2013] Bergstra, J., Yamins, D., and Cox, D. D. (2013). Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20.

[Blum et al., 2012] Blum, M., Springenberg, J. T., Wülfing, J., and Riedmiller, M. (2012). A learned feature descriptor for object recognition in rgb-d data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1298–1303. IEEE.

[Bo et al., 2011a] Bo, L., Ren, X., and Fox, D. (2011a). Depth kernel descriptors for object recognition. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 821–826. IEEE.

[Bo et al., 2011b] Bo, L., Ren, X., and Fox, D. (2011b). Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In *Advances in neural information processing systems*, pages 2115–2123.

[Bo et al., 2013] Bo, L., Ren, X., and Fox, D. (2013). Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402. Springer.

[Browatzki et al., 2011] Browatzki, B., Fischer, J., Graf, B., Bülthoff, H. H., and Wallraven, C. (2011). Going into depth: Evaluating 2d and 3d cues for object classification on a new, large-scale object dataset. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1189–1195. IEEE.

[Cheng et al., 2015] Cheng, Y., Zhao, X., Huang, K., and Tan, T. (2015). Semi-supervised learning and feature evaluation for rgb-d object recognition. *Computer Vision and Image Understanding*, 139:149–160.

[Coates et al., 2011] Coates, A., Ng, A. Y., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

[Eitel et al., 2015] Eitel, A., Springenberg, J. T., Spinello, L., Riedmiller, M., and Burgard, W. (2015). Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE.

[Farabet et al., 2013] Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

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

[Hyvärinen and Oja, 2000] Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430.

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

[Jhuo et al., 2014] Jhuo, I.-H., Gao, S., Zhuang, L., Lee, D., and Ma, Y. (2014). Unsupervised feature learning for rgb-d image classification. In *Computer Vision–ACCV 2014*, pages 276–289. Springer.

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

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Lai et al., 2011] Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE.

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

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

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

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

[Payan and Montana, 2015] Payan, A. and Montana, G. (2015). Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. *arXiv preprint arXiv:1502.02506*.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[Salakhutdinov and Hinton, 2009] Salakhutdinov, R. and Hinton, G. E. (2009). Deep

boltzmann machines. In *International conference on artificial intelligence and statistics*, pages 448–455.

[Schwarz et al., 2015] Schwarz, M., Schulz, H., and Behnke, S. (2015). Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1329–1335. IEEE.

[Scovanner et al., 2007] Scovanner, P., Ali, S., and Shah, M. (2007). A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia*, pages 357–360. ACM.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[Socher et al., 2012] Socher, R., Huval, B., Bath, B., Manning, C. D., and Ng, A. Y. (2012). Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673.

[Song and Xiao, 2015] Song, S. and Xiao, J. (2015). Deep sliding shapes for amodal 3d object detection in rgb-d images. *arXiv preprint arXiv:1511.02300*.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Taigman et al., 2014] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708.

[Tombari et al., 2010] Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pages 356–369. Springer.

[Tombari et al., 2011] Tombari, F., Salti, S., and Stefano, L. D. (2011). A combined texture-shape descriptor for enhanced 3d feature matching. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 809–812. IEEE.

[Toshev and Szegedy, 2014] Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660.

[Wang et al., 2015] Wang, A., Lu, J., Cai, J., Cham, T.-J., and Wang, G. (2015). Large-margin multi-modal deep learning for rgb-d object recognition. *Multimedia, IEEE Transactions on*, 17(11):1887–1898.

[Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328.