RGB-D Object Recognition Using Deep Convolutional Neural Networks

BMVC 2017 Submission # 361

Abstract

We address the problem of object recognition from RGB-D images using deep convolutional neural networks (CNNs). We advocate the use of 3D CNNs to fully exploit the 3D spatial information in depth images as well as the use of pretrained 2D CNNs to learn features from RGB-D images. There exists currently no large scale dataset available comprising depth information as compared to those for RGB data. Hence transfer learning from 2D source data is key to be able to train deep 3D CNNs. To this end, we propose a hybrid 2D/3D convolutional neural network that can be initialized with pretrained 2D CNNs and can then be trained over a relatively small RGB-D dataset. We conduct experiments on the Washington dataset involving RGB-D images of small household objects. Our experiments show that the features learnt from this hybrid structure, when fused with the features learnt from depth-only and RGB-only architectures, outperform the state of the art on RGB-D category recognition.

1 Introduction

Object recognition is a fundamental problem with numerous applications in computer vision and robotics. With easy availability of low-cost sensors like Microsoft Kinect, depth and color information can be simultaneously captured and included in recognition of objects. Depth provides additional information about the 3D structure of the physical environment and has proven to improve the recognition performance when paired with color information. Unlike RGB images, depth images are invariant to lighting and allow better background separation.

Object recognition and classification have been extensively studied for RGB images, and there are large datasets available. Convolutional Neural Networks (CNNs) have been particularly successful and have produced state of the art results on these large datasets in challenges like ImageNet [17]. The ImageNet challenge has led to development of successful deep CNNs for image classification like AlexNet [11], VGGnet [19], GoogleNet [22] and ResNet [9]. 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 is in general time-consuming and requires large amount of processing time while training. Likewise, 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 hand-designed feature descriptors. In this work, we utilize deep convolutional neural networks pretrained on a large RGB dataset and address the problem of transfer learning from 2D source to 3D for object recognition on relatively small RGB-D datasets containing 3D information. In particular, we conduct experiments on the RGB-D Washington dataset involving household objects [12].

We present an approach that exploits the RGB information learnt by large scale models, particularly the VGGnet, so as to train a novel hybrid 2D/3D convolutional neural network and boost the recognition performance via fusion. Our contributions include:

1. We exploit the information in the pretrained VGGnet model to extract features from RGB images and train a linear SVM (Support Vector Machine) with these features for RGB-based category recognition. Our approach exceeds the state-of-the-art on the Washington dataset.

2. We study the problem of training 3D convolutional neural networks from scratch based on RGB-D images. We preprocess the depth information to produce a spatial 3D voxel representation combining depth and RGB information.

3. We modify the VGGnet in such a way that it can accept 3D inputs and after the first layer it continues as 2D like the original VGGnet. By modifying the first layer of the VGGnet, we can initialize the resulting 2D/3D hybrid network, that we refer to as VGG3D, by transferring the weights from the original VGGnet.

4. Finally, we fuse the features resulting separately from VGGnet, 3D CNN and VGG3D architectures. Our fusion results exceed the state-of-the-art for category recognition.

2 Related Work

The previous methods proposed for RGB-D objected recognition are broadly divided into two categories: the methods that use hand-designed descriptors and those that learn features. These features are then fed into classifiers along with their labels for the final classification task which is usually based on SVMs or softmax regression. Some of the methods that use feature learning include sparse coding, hierarchical matching pursuit [3, 4], convolutional k-means descriptors [2], regularized reconstructed ICA network [10] and coupled classifiers [14].

In recent years, deep learning has become extremely popular and has been extensively applied to machine learning tasks. In particular, convolutional neural networks are being popularly used to solve vision related task such as scene labeling [7], object recognition [20], face verification [23] and pose estimation [25]. Convolutional neural networks were originally introduced by [13] for a hand written digit recognition problem. Since then they have been used on datasets that include rich images like ImageNet and have achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge [17]. Recently, application of convolutional neural networks for RGB-D data has become popular and various methods have been suggested to achieve superior performance as compared to hand-designed descriptors.

Convolutional neural networks have been used in combination with other architectures to solve the RGB-D object recognition problem. One such technique is a combination of convolutional and recursive neural networks, that is based on the idea that convolutional layers extract low level features and recursive neural networks extract high level features [20]. Another work modifies this technique to boost the RGB-D based recognition performance by proposing a semi-supervised framework based on co-training, which uses less labeled data but achieves competitive results as compared to the state of the art [5]. A recent work tackling the same problem of joint learning introduces a multimodal layer to a CNN-based neural network [26]. Another interesting work [18] converts depth images to RGB images using a color map and then uses a deep convolutional neural network (AlexNet) pretrained on color images. An extension of this method further improves the recognition performance by introducing a multi-modal scheme to learn joint features from the pretrained AlexNet [6].

Due to the availability of faster GPUs and dedicated CUDA libraries for deep learning, 3D convolutional neural networks are now increasingly becoming popular. 3D CNNs are currently being used for region proposal, object recognition and medical imaging. One such work focuses on neuroimaging using 3D MRI scans to predict Alzheimer's disease. The network consist of 3D convolutional layers pretrained via unsupervised learning using sparse auto-encoders [16]. Several approaches have been suggested to get a 3D voxel representation from depth images to be fed into 3D CNNs that allow to better exploit the 3D structure present in depth images. One such approach, called ShapeNets, represents the depth information into a voxel grid in the form of truncated signed distance function (TSDF) [21]. VoxNets is another approach that encodes depth information into a volumetric occupancy grid [15].

Transfer learning is a technique which improves the learning on target task using the information gathered on source task [24]. Especially in the case of object recognition, transfer learning is widely used with deep convolutional neural networks. The most common strategy is to use a deep CNN architecture pretrained on a large dataset as a feature extractor [21], [18] or to fine-tune it on a smaller dataset [6], [26]. When the target dataset is small, using a network that is pretrained with a larger dataset shows better performance on object recognition as investigated in [27]. This strategy of transfer learning however is currently applicable, particularly in the case of object recognition, only when the source and target datasets are of the same type, i.e., involving purely 2D image data. Transfer learning from 2D source to 3D target remains to be an open problem that we attempt to tackle in this work.

3 The Dataset

We conduct experiments on the Washington RGB-D dataset [12]. The dataset consists of 300 small household objects such as fruits, vegetables, boxes and water bottle, which are instances of 51 categories. Hence the dataset is grouped by category as well as by instance. For example, apple is a category that has five instances each of which can be red, green or yellow. Each instance has depth and RGB images from three video sequences. Each 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 captured with the camera at 30°, 45°, and 60°. In this work, we use the cropped version of the dataset, which consists of bounding box images, along with the segmentation masks that filter the background from both depth and RGB images. Our evaluation is based on a subset of this dataset, which consists of every fifth frame of each of the video sequences, resulting in about 42000 images. Out of these images, around 35000 instances are used for training and 7000 for testing.

The Washington dataset can be considered as tricky for the object recognition task due to various reasons. For example, the ball object can have instances that are not only of different colors but also of different shapes and sizes. 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 indeed hard to differentiate.

4 VGGnet for RGB-only recognition

For RGB-only object recognition, we extract features using the pretrained VGGnet [19], which was the runner-up architecture in the ImageNet [17] classification and localization challenge in 2014. The VGGnet has a deep but simple structure, deploying very small (3×3) filters in all of its convolutional layers. Hence it is relatively easy to implement and train, and therefore commonly used in computer vision for feature extraction from RGB images. More specifically, we use the 16-layer VGGnet which consists of stacks of convolution layers followed by maxpooling layers. In this deep architecture, the feature maps increase in number as the depth increases, and the convolutional layers are eventually followed by three fully connected layers.

While the initial layers of the VGGnet learn general features, deeper layers are expected to learn more dataset specific features. To extract the most meaningful features and make best use of the VGGnet for our task, we investigate the last three fully connected layers of the VGGnet to understand where the network inclines towards learning more dataset specific features. For this, we remove the first, second, and third fully connected layers from the VGGnet respectively (or their combinations) and extract features for each image by applying a forward pass on the pretrained VGGnet with no fine-tuning. The resulting features are then used for training a linear SVM in each case. Based on the experimental results that we will later present in Section 5, we use the VGGnet up to the depth of its first fully connected layer as the final architecture, as shown in Figure 1, where “Conv" stands for 3×3 convolution followed by ReLU activation, "Pool" for max-pooling and "FC" for fully connected layer.

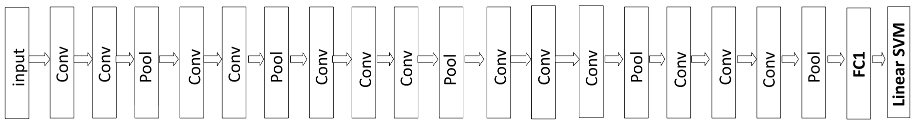

Figure 1: Our pretrained VGGnet-based architecture for RGB object recognition.

5 3D Pipeline

5.1 3D Input Representation

The first step in our 3D recognition pipeline is to convert the input depth information into an adequate 3D representation. Rather than encoding the depth information as any function or descriptor such as truncated signed distance function as in [21], we represent the depth information as raw as possible along with RGB information and investigate if a 3D CNN can learn meaningful features.

We represent the depth information in a 3D voxel grid by defining a third dimension based on the depth values present in the RGB-D images of the dataset. We create a 3D voxel representation, with the same height and width as the original image, and with a depth determined by the difference between the maximum and minimum depth values found in the images. Each RGB-D pixel of an image is then placed at the same position in the voxel grid but at its corresponding depth. This results in a 3D representation that simultaneously encodes the 3D spatial and color information of a given object. Incorporating the RGB information into the 3D representation helps to jointly learn features that are related to both depth and color rather than learning features from depth only. Our voxel representation (only for R channel for simplicity) is shown in Figure 2.

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. We then apply the provided segmentation mask to filter out the background information and encode only the object shape since the turntable in the background has similar depth values to the object and interfere with its shape.

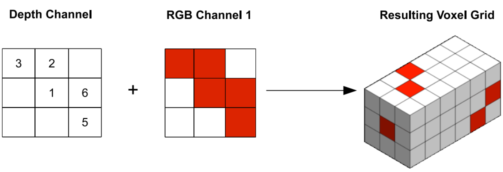

Figure 2: Illustration of how RGB-D images are converted to voxel representations for a 3×3 input image (fragment), where depth values are quantized into 6 intervals.

5.2 3D CNN Architecture

We follow the VGGnet [19] in terms of architecture while designing our 3D CNN. We employ two convolutional layers, each followed by a max-pooling layer. Two fully connected layers are added to the end of the network, as shown in Figure 3. We rescale the resolution of the voxel representation to $30 \times 30 \times 30$, which is considerably smaller than the resolution of the original RGB-D images, but sufficient to train the network so as to obtain a decent performance. Note also that training 3D CNNs is significantly more demanding in terms of computation and memory requirements when compared to 2D CNNs. We use 64 filters at each convolutional layer and keep the filter size small (3×3). The number of filters is maintained through the convolution layers to retain information since the input size is already smaller as compared to the original object size. The weights are randomly initialized using the Xavier technique [8] and the network is trained by backpropagation using softmax classifier.

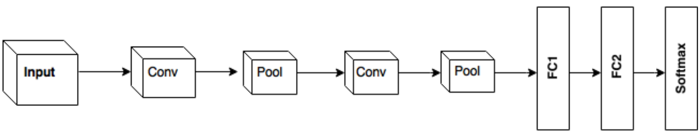

Figure 3: Our 3D CNN architecture

5.3 Hybrid CNN Architecture (VGG3D)

In order to transfer learning from 2D source to 3D target, we define a VGGnet-like hybrid CNN structure than can be initialized with the VGGnet and trained with backpropagation over the 3D RGB-voxel input generated as described in Section 5.1. The main idea is to modify the 2D VGGnet into a network that can accept 3D information. To this end, we replace the first layer of VGGnet-16 with a 3D convolutional layer by adding a dimension to the pretrained filter weights. After the first layer, the resulting hybrid network continues as 2D like the original VGGnet, and produces the same result as the original VGGnet would generate when fed with the corresponding RGB input image. This provides us with a good starting point to fine-tune the weights transferred from the VGGnet. We call this modified network as VGG3D and visualize its structure in Figure 4.

In our experiments with the hybrid network, we encode the depth values via non-uniform quantization based on the distribution of pixels along the third dimension that we denote by d, resulting in a $N \times N \times D \times 3$ voxel representation, where D denotes the depth of the voxel grid and 3 is the number of color channels. We set $N = 224$, which is the image size given as input to the VGGnet. The depth values are quantized into $D-1$ intervals of varying length so as to include equal number of pixels (points) at each interval over the whole dataset. The last depth interval D is spared to the depth values corresponding to the background. We compute background depth values using the inverse of the segmentation masks provided. We choose $D = 6$ as an optimal value in terms of memory constraints and the performance it gives.

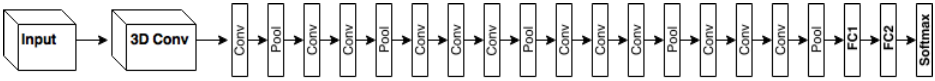

Figure 4: Our Hybrid 2D/3D CNN architecture (VGG3D)

The filters at the first layer of the hybrid network are 3D convolution kernels of size $3 \times 3 \times D$. The weights of these filters are initialized by replicating the filters of the first layer of the VGGnet along the depth dimension d. The filters at the remaining layers all remain 2D, initialized directly with the weights of the corresponding layers of the VGGnet.

In the sequel, we explain more rigorously how we initialize the hybrid CNN so that the output of the modified first layer generates exactly the same output as the first layer of the VGGnet when fed with the same sample (RGB or RGB-voxel). Let $x^{(2)}(i,j)$ denote the 2D input image of size $N \times N$ and $x^{(3)}(i,j,d)$ the input 3D voxel grid of size $N \times N \times D$. For simplicity, we assume that the input images are monochrome with single channel, but the analysis can easily be generalized to RGB images. The kth filter at the first layer of the VGG3D, denoted by $w_k^{(3)}(i,j,d)$ of size $3 \times 3 \times D$, is generated by replicating the corresponding 2D filter of the VGGnet along dimension d so that

$$w_k^{(3)}(i,j,d) = w_k^{(2)}(i,j), \tag{1}$$

where $w_k^{(2)}(i,j)$ is the kth filter at the first layer of the VGGnet, which is of size 3×3. The corresponding outputs at the first layers of the VGGnet and VGG3D are then given by $y_k^{(2)}(i,j) = x^{(2)}(i,j) * w_k^{(2)}(i,j)$ and $y_k^{(3)}(i,j,d) = x^{(3)}(i,j,d) * w_k^{(3)}(i,j,d)$, respectively. By Eq. 1, we can then write

$$y_k^{(3)}(i,j,D/2) = y_k^{(2)}(i,j) \tag{2}$$

assuming D is even, since the 3D input is originally a depth image so that $x^{(3)}(i,j,d)$ is non-zero for at most one value of d, i.e., only one voxel is occupied for a given pixel (i,j). When the output $y_k^{(3)}(i,j,D/2)$ of the first layer of the VGG3D is then fed to the next layer, the VGG3D generates exactly the same final output as the VGGnet would produce with the same sample. The illustration of this process is shown in Figure 5.

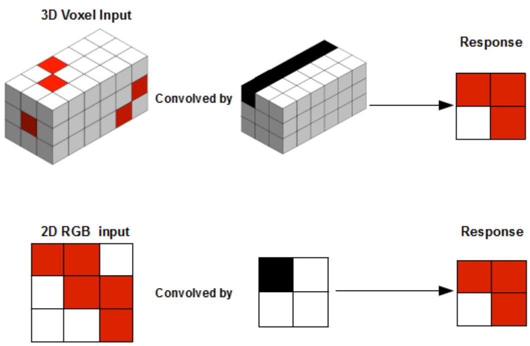

Figure 5: Illustration of VGG3D (top row) and VGGnet (bottom row) first layer responses.

6 Fusion

We fuse the outputs of the pretrained VGGnet, the 3D CNN and the VGG3D architectures to get our final overall RGB-D recognition performance. We basically concatenate the features that we get from individual architectures and then feed the resulting vector to the linear SVM. Although the 3D voxel input already contains RGB information, the 3D CNN is trained on a much lower resolution than the VGGnet resulting in a loss of RGB information. So we do not expect it to model the RGB information as good as the VGGnet does. But incorporation of RGB information into the 3D CNN helps to train it from scratch using random initialization on a smaller dataset and to contribute to the overall performance by exploiting mainly the depth information. Moreover, the inclusion of the VGG3D in the fusion is expected to compensate some of the 3D information that cannot be modeled by the 3D CNN due to difficulties in its training, and the VGG3D achieves this via transfer learning.

7 Experiments

7.1 Setup

We implement all the networks using Julia programming language [1] and Knet framework [28]. The experiments are carried out for the category recognition problem over 10 splits as in [12]. Each split contains randomly selected objects from each category (51 in total) in the test set and the remaining 249 objects are included in the training. The reported performance results are all outputs of the linear SVM fed by the features resulting from individual architectures or their combinations. Validation and parameter optimization, regarding both SVM classifiers and CNN architectures that we employ, are performed on Split 1 and repeated with fixed settings over the remaining splits.

For 3D CNN training, we fix the learning rate to 0.1 and the number of epochs to 10. The network is trained using back-propagation. For VGG3D network training, the softmax layer is first trained with back-propagation by freezing all the other layers with the weights transferred from the pretrained VGGnet. This learnt layer is then used to initialize the softmax layer prior to training of the VGG3D as a whole with backpropagation. For VGG3D training, we choose a low learning rate of 0.0001, which helps prevent overfitting.

Prior to the experiments, all the images in the Washington dataset are re-scaled to the input size of the original VGGnet (224×224) and then mean-normalized (the mean is computed over the training set). Beside this, no other preprocessing is applied.

7.2 Results

We first compare the performances of the features extracted from different layers of the VGGnet in Table 1 (see also Section 4). We observe that as we move closer to the last fully connected layer of the VGGnet, the performance significantly worsens. This is because these layers learn features specific to the object categories of the original dataset. The first fully connected layer performs the best as anticipated. We also consider fusing the features resulting from a combination of fully connected layers as in [18]. We observe that when the best performing layers are fused, we get only an increase of 0.03% in the performance. Since the feature size doubles while fusing, we decide to include only the best performing layer in our VGGnet based architecture for RGB recognition (see Figure 1).

VGG Layer	Feature Size	Accuracy (%)
FC1	4096	91.08
FC2	4096	89.25
FC3	1000	72.24
FC1+FC2	8192	91.11
FC2+FC3	5096	89.20

Table 1: Recognition performance results with features extracted from fully connected VGGnet layers over Split1, where FC-n denotes the nth fully connected layer.

Table 2 shows the 10-fold recognition results for VGGnet, 3D CNN and VGG3D architectures, and their combinations, along with the corresponding mean accuracies and standard deviations. The VGGnet shows good performance of around 89% recognition rate on RGB images while the 3D CNN performs around 78% with incorporation of depth. However, the 3D CNN adds a significant 2.5% boost to the recognition performance when fused with the VGGnet. There are a number of reasons why the 3D voxel input, although comprising both depth and RGB information, does not yield better results than the RGB-only data itself. First, unlike the VGGnet, the 3D CNN is trained from scratch via random initialization and thus can not take any advantage of any previously learnt information. Second, the Washington dataset is a small dataset when compared to the ImageNet RGB database on which the VGGnet was pretrained. Moreover, the RGB-D images in the Washington dataset need to be re-scaled into a small voxel resolution in order to train the 3D CNN structure, which inevitably yields loss of both RGB and depth information. But when fused with the VGGnet, the 3D CNN that jointly learns information from depth and RGB adds significantly to the performance. When the VGG3D is finally incorporated into the fusion scheme, the VGG3D compensates for some part of the loss in 3D information via transfer learning, and the overall performance is further boosted and exceeds the state of the art, as given in Table 3, which presents 10-fold recognition results in comparison to previous methods. Note also that the performance of our VGG3D network is superior to the individual performances of the VGGnet and the 3DCNN. To the best of our knowledge, our overall fusion scheme achieves the highest accuracy on category recognition compared to the previous methods tested on the Washington dataset.

Split	VGGnet	3D CNN	VGG3D	VGGnet + 3D CNN	VGGnet + 3D CNN + VGG3D
1	91.04	76.33	91.22	91.03	91.90
2	92.69	76.88	92.51	92.09	92.76
3	86.15	79.96	87.88	90.90	91.69
4	87.62	74.69	87.56	90.27	90.31
5	88.84	78.63	89.53	92.39	92.63
6	89.72	79.61	90.02	90.40	91.02
7	90.70	83.12	90.92	92.57	92.82
8	87.87	77.40	88.32	91.64	92.27
9	88.79	77.40	89.91	90.35	90.76
10	86.15	80.30	89.97	91.56	92.21
Mean	**88.96**	**78.43**	**89.78**	**91.29**	**91.84**
Dev	2.13	2.41	1.55	0.86	0.89

Table 2: Accuracy results (%) with 10-fold split validation for VGGnet, 3D CNN, VGGnet and fusion combinations.

Method	RGB	Depth	RGB-D
[12]	74.30 ± 3.3	53.1 ± 1.7	81.90 ± 2.8
[4]	82.40 ± 3.1	81.2 ± 2.3	87.50 ± 2.9
[18]	83.10 ± 2.0	N/A	89.40 ± 1.3
[20]	80.80 ±	78.90 ± 3.8	86.80 ± 3.3
[10]	85.65 ± 2.7	**83.94 ± 2.8**	89.59 ± 3.8
[5]	85.20 ±1.2	81.2 ± 2.3	90.10 ± 1.1
[6]	84.10 ± 2.7	83.8 ± 2.7	91.30± 1.4
[26]	74.6 ± 2.7	N/A	86.90 ± 2.9
Ours	**88.96 ± 2.1**	78.43 ± 2.4	**91.84 ± 0.89**

Table 3: Comparison of our fusion scheme with previous methods for category recognition: Accuracy results (%) with RGB-only, Depth-only and RGB-D modalities.

8 Conclusion

This work can be considered as an attempt to transfer learning from 2D source to 3D target for the object recognition problem where the datasets comprising 3D information are not large enough to be able to train deep neural network architectures from scratch. Our findings show that explicit handling of 3D spatial information as an additional modality to RGB data (using 3D CNNs in our case) significantly contributes to the recognition performance. Moreover, even a small amount of learning transferred from 2D source data to 3D (in terms of the first layer of the VGGnet transferred to the VGG3D in our case) can help further boost the performance beyond the state of the art. We believe that there is still room for even further improvements and hence need for better and more comprehensive architectures that can be trained via transfer learning in order to fully exploit the 3D information available for object recognition and possibly for other multimodal vision tasks as well.

References

[1] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[2] M. Bluml, J. T. Springenberg, J. Wülfing, and M. Riedmiller. 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, 2012.

[3] L. Bo, X. Ren, and D. Fox. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In *NIPS*, volume 1, page 6, 2011.

[4] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013.

[5] Y. Cheng, X. Zhao, K. Huang, and T. Tan. Semi-supervised learning and feature evaluation for RGB-D object recognition. *Computer Vision and Image Understanding*, 139: 149–160, 2015.

[6] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust RGB-D object recognition. In *2015 IEEE/RSJ International Conference on*, pages 681–687, 2015.

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

[8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

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

[10] I. Jhuo, S. Gao, L. Zhuang, D. T. Lee, and Y. Ma. Unsupervised feature learning for RGB-D image classification. In *Asian Conference on Computer Vision*, pages 276–289. Springer, 2014.

[11] A. Krizhevsky, I. Sutskever, and G. H. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[12] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2011.

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

[14] X. Li, M. Fang, J. Zhang, and J. Wu. Learning coupled classifiers with RGB images for RGB-D object recognition. *Pattern Recognition*, 61:433–446, 2017.

[15] D. Maturana and S. Scherer. 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, 2015.

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

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

[18] M. Schwarz, H. Schulz, and S. Behnke. 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, 2015.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3D object classification. In *NIPS*, volume 3, page 8, 2012.

[21] S. Song and J. Xiao. Deep sliding shapes for amodal 3D object detection in RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, Vincent V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[23] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. 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, 2014.

[24] L. Torrey and J. Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

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

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

[27] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014.

[28] D. Yuret. Knet: beginning deep learning with 100 lines of julia. In *Machine Learning Systems Workshop at NIPS 2016*, 2016.