Deep Learning for Single-View Instance Recognition

David Held, Sebastian Thrun, Silvio Savarese
Stanford University
{davheld, thrun, ssilvio}@cs.stanford.edu

Abstract

Deep learning methods have typically been trained on large datasets in which many training examples are available. However, many real-world product datasets have only a small number of images available for each product. We explore the use of deep learning methods for recognizing object instances when we have only a single training example per class. We show that feedforward neural networks outperform state-of-the-art methods for recognizing objects from novel viewpoints even when trained from just a single image per object. To further improve our performance on this task, we propose to take advantage of a supplementary dataset in which we observe a separate set of objects from multiple viewpoints. We introduce a new approach for training deep learning methods for instance recognition with limited training data, in which we use an auxiliary multi-view dataset to train our network to be robust to viewpoint changes. We find that this approach leads to a more robust classifier for recognizing objects from novel viewpoints, outperforming previous state-of-the-art approaches including keypoint-matching, template-based techniques, and sparse coding.

1. Introduction

There are many real-world scenarios in which we want to recognize an object instance from just a single training example. For example, for many product databases available on Amazon, Safeway, or other websites, only a small number of images are available for each product. Given a novel viewpoint of a product, can we robustly recognize the target object?

Enabling a computer vision system to recognize objects from just one training example would enable a range of applications that could train on images from product databases. For example, a kitchen perception system might need to recognize the grocery products in the kitchen. Such a system would ideally be trained from a grocery product database, even if only one image of each product were available.

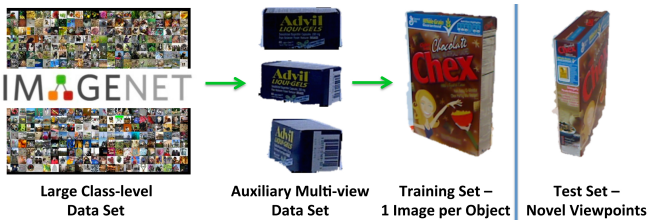

Figure 1. Given only a single image of an object, we want to recognize this object from novel viewpoints. We perform a multi-stage training procedure, in which we first pre-train on a large class-level dataset, followed by an auxiliary multi-view dataset, which trains our network to be robust to viewpoint changes. Finally we train on the objects we wish to recognize from just a single image.

We would also like to enable casual users to train a classifier to recognize an object after taking just a single picture of the target object. Such a method could be used to bootstrap a number of custom applications that require understanding how a user interacts with the objects in their environment. Thus, we need computer vision methods that can robustly recognize objects after training from just a single image.

Traditionally, researchers have used feature-matching based approaches to recognize objects from a single example. Unfortunately, because feature-matching approaches rely on being able to detect distinctive keypoints on an object, they often fail for textureless objects or for non-planar objects with large changes in viewpoint [41]. Machine learning methods, including deep learning, have been successfully applied to recognize objects at the class level [27, 18], but they are not commonly used to recognize specific object instances, especially when only a single training image is available for each object.

We introduce a new approach to training neural networks to recognize objects from just a single image, using a general-to-specific training procedure. We initially use a large dataset to train our network to recognize general object classes. We then train our network on a smaller dataset in which we observe objects from multiple viewpoints. Finally, we train our network on a separate dataset in which only a single image is available for each object instance, as

depicted in Figure 1.

By training our network in this general-to-specific manner, our network learns the invariances that it needs to perform the final task. Our network initially learns general visual properties about the world. It then learns generic object invariances, enabling the network to be robust to rotations and changes in viewpoint. Finally, our network learns to recognize a specific set of objects from just a single training image per object.

Using this novel multi-stage training procedure, our network learns to robustly recognize objects from new viewpoints. To our knowledge, this is the first work that uses deep learning to recognize specific object instances from a single image. We perform an extensive evaluation and show that multi-view pre-training outperforms previous state-of-the-art approaches for recognizing both textured and untextured objects from novel viewpoints.

2. Related Work

Instance recognition has traditionally been achieved using either keypoint-based methods [38, 3] or by matching local image patches [14, 45, 31]. Keypoints can be filtered using different criteria [38] and validated using RANSAC or Hough Voting to ensure geometric consistency [15]. Although keypoint-based approaches have shown some success for image recognition [9], such methods are unreliable for recognizing untextured objects or non-planar objects when the viewpoint is changed by more than 25 degrees [41].

Template matching has also been used for instance recognition [24, 42]. Much work has recently been done to make template matching scalable, efficient, and robust to occlusions [20, 23, 11]. However, viewpoint invariance is usually achieved by recording many templates during training from different viewing angles. If only a small number of images are available from each object during training, template matching methods will not robustly detect the target object, as we will demonstrate.

Another approach that can be used for recognizing objects is to use machine learning methods to train an object classifier [32, 7]. One example of such a classifier that has shown great success in recent years is a convolutional neural network [16, 33, 27]. However, statistical methods such as neural networks typically require many training examples to perform well. For example, for the ImageNet challenge, participants train their methods on 1.2 million training examples [12]. Recently, some groups have successfully trained their networks on just 5000 images across 20 classes [1], sometimes using domain-specific fine-tuning [18]. We will test the performance of neural networks when only 1 training example is available per class.

One-shot learning has also been explored for classifying objects at the category-level [13, 51, 47] or for recognizing handwritten characters [29, 30]. In constrast, we focus on recognizing object instances from novel viewpoints, and we compare our approach to state of the art techniques for object instance recognition.

Our method makes use of a separate multi-view dataset to improve performance on the task of instance recognition from a single training image. Our idea of using a supplemental multi-view dataset is related to previous efforts to improve recognition performance by using a video sequence [40, 4]. Another related effort is to use unlabeled video for unsupervised feature learning [34, 53]. These methods typically enforce the consistency of features between subsequent video frames. We instead use multi-view objects in a classification setting to improve our performance for recognizing single-view objects, and we do not treat the multi-view dataset as a linear video sequence.

Some researchers have attempted to measure the invariance of deep networks to rotations and other types of transformations [19, 43, 35]. These papers have focused on measuring the rotational invariance for image patches or for general object classes rather than object instances.

The problem of adapting a class-level classifier for use in viewpoint-invariant instance recognition is also related to the topic of domain adaptation [5, 22, 21, 17]. However, in most domain adaptation problems, the source dataset contains the same object labels as during inference. The goal of domain adaptation is to adapt a classifier trained from a given source domain (e.g. Amazon images) to classify the same set of objects in the target domain (e.g. Webcam images). On the other hand, for our task the classifier must learn to recognize novel object instances, with no overlap between our large "source" datasets and the final query images that we wish to recognize. Further, we are interested in adapting class-level classifiers for viewpoint invariant instance recognition, which has not been explored previously.

3. Method

3.1. Problem Setup

Suppose that we are given an image x_i of an object instance that we want to recognize. We assume that we have a "single-view" database of K_S different objects, and that the object in our image x_i is one of the K_S objects in our single-view database. We also assume that each of the objects in our database has only one image taken of it. Given that our image x_i is likely to be taken from a novel viewpoint relative to the images in our database, how can we robustly identify the instance label for this object?

In order to robustly perform this task, we suppose that we also have a separate "multi-view" set of K_M objects for which we have recorded images from many viewpoints. Because we have observed each of these separate objects from many viewing angles, we can use these images to teach our

method to be invariant to viewpoint changes. Then, given a novel viewpoint of an object from the single-view dataset, we can use this learned invariance to correctly recognize the target object.

Note that the multi-view objects are chosen so that there is no overlap between the K_M multi-view objects and the K_S single-view objects. Thus, any invariances that we learn from the multi-view dataset must be general to be able to transfer over to a new set of objects. Our final goal is to identify an image x_i as belonging to one of the K_S single-view objects; the multi-view dataset is helpful only in teaching our method to be invariant to viewpoint changes.

3.2. Multi-View Pre-Training

We consider instance recognition as a classification problem, and we will explore the use of neural networks to perform this task. Because neural networks represent a non-convex decision boundary, the initialization of the network is important. One common approach for training a neural network with a limited amount of data is to initialize the network by pre-training on a larger dataset [18] (*e.g.* ImageNet [12]). These initial weights are then fine-tuned using a smaller dataset for the relevant task. This training procedure allows the network to find a better local optimum.

However, the ability to transfer information from the larger dataset to the smaller dataset, via network initialization, depends on the similarity between the datasets. If the datasets are not very similar, then this initialization will be poor [54]. As we will show, pre-training the network for class-level recognition (*e.g.* using ImageNet) is not ideal for training these networks to be viewpoint invariant with respect to specific object instances.

For the original ImageNet classification task, the goal of the network is to recognize 1000 different object classes. Each class represents an object category, such as "restaurant" or "mask," and the appearance of objects within the class can vary dramatically; different restaurants can have a very different appearance. Because the network must recognize generic object classes, the computational effort of the network is spent attempting to handle all of the different aspects of intra-class variability. On the other hand, if our goal is to perform object instance recognition, then we can focus our network's computational effort on being robust to rotations, leading to better performance at this task.

We will show that, although pre-training our network on ImageNet provides a decent initialization for our network, we can obtain better performance through a multi-stage training procedure, as follows:

1. Train our network on a large class-level dataset.
2. Train our network on an instance-level dataset with many views per object instance.
3. Train our network to recognize a new set of object instances from a single image per object.

This setup is illustrated in Figure 1. In more detail, we initially pre-train our network on a large class-level dataset, *e.g.* ImageNet, which allows our network to learn general image statistics. We then train our network on a smaller dataset in which we observe a set of objects from multiple viewpoints, and we learn to recognize these objects instances. This stage allows our network to learn to be robust to changes in viewpoint. Finally, we train our network on a separate dataset in which only a single image is available for each object. We show that adding an intermediate multi-view pre-training step (step 2 above) gives better performance than pre-training only on a class-level dataset. Adding multi-view pre-training increases the robustness of our network and enables us to recognize novel objects from new viewpoints.

We would also like to be able to recognize objects in real scenes against random backgrounds. To make our network robust to different backgrounds, during multi-view pre-training (step 2) we synthetically place the objects against random background scenes which do not contain any of the test objects. Although the single-view objects that we wish to recognize are placed against a fixed background for training (in step 3), we will show that pre-training with separate multi-view objects in step 2 against random backgrounds allows our method to learn to be robust to new backgrounds.

One can view our approach as an extension of data augmentation techniques for neural networks. It is common when training neural networks to perform multiple image transformations on each training example to synthetically generate more training examples. Common transformations include crops, horizontal flips, and lighting changes [27].

These data augmentation methods are an attempt to train the network to be robust to translations or changes in lighting. However, it is more difficult to construct an image transformation that would simulate an out-of-plane rotation. As an alternative, we propose multi-view pre-training, in which our intermediate stage involves classifying a separate set of objects when trained from multiple viewpoints. Multi-view pre-training allows our network to learn new kinds of invariances, such as out-of-plane rotations, that would be hard to simulate using data augmentations.

3.3. Network Details

Our neural network uses the CaffeNet architecture [25], which is very similar to the architecture proposed by Krizhevsky *et al*. [27]. The network is initially pre-trained on ImageNet [12]. We then fine-tune this network on the multi-view dataset as follows: we replace the final layer with a K_M class classifier, and we fine-tune the weights to classify the K_M multi-view objects. We call this step "multi-view pre-training" since we are training the network to recognize object instances given multiple views of each object. During multi-view pre-training, we hold the convo-

lutional layers fixed and only fine-tune the fully-connected layers on top. The number of layers that we fix was determined using a hold-out validation set.

During multi-view pre-training, we use a learning rate of 0.001 for all layers except the final layer, which we set to a learning rate of 0.01. After 50,000 iterations, we reduce the learning rate by a factor of 10, and after 100,000 iterations we stop the multi-view training. Other hyperparameters are taken from the default parameters for CaffeNet [25], and we left them unchanged.

Finally, we initialize the network using the learned weights from multi-view pre-training, and we fine-tune the network to classify the single-view objects. To do this, we replace the final layer with a K_S class classifier for the K_S single-view objects. Each object in this dataset has only 1 training example from a single viewpoint. We use the same parameters as before, except that the learning rates are reduced by a factor of 10, which was again determined using cross-validation on a hold-out set. The final classifier is used to classify these K_S objects from novel viewpoints. We call a classifier trained in this manner a "neural network with multi-view pre-training."

4. Results

We perform a number of experiments to analyze the performance of different instance recognition methods. In Sections 4.1 through 4.3, we use the RGB-D object dataset [28], in which we recognize objects that are placed on a turntable and recorded from different viewpoints. In this controlled setup, we can measure the object's angular difference between the training and test images, allowing us to compute how robust the different methods are to out-of-plane rotations. We will later evaluate the methods on recognition in real-world scenes, as described below.

We evaluate the performance of different methods on this dataset under three conditions:

1. Training from many examples (Section 4.1)
2. Training from a variable number of examples (Section 4.2)
3. Training from just a single example (Section 4.3)

In all three cases, we use the same test set, which is the instance recognition test set from [28]. None of the methods that were evaluated use depth information except to segment out the target object.

We vary the number of examples available during training to show how well each method generalizes with a limited number of training examples. When multiple training examples are available, we compare the neural network-based approaches to other machine learning approaches. When only one training example is available, we also evaluate keypoint-matching and other approaches that are designed to match pairs of images. We find that neural networks have superior performance in all three cases, and we further show the advantage of multi-view pre-training in the case of training from just a single example.

Finally, in Section 4.4, we evaluate how robust the different classifiers are to handling occlusions and real backgrounds. For this we use the RGB-D scenes dataset [28], in which the objects from the previous test set are placed in a real-world scene. Our task now is to recognize the object given the object's bounding box. In an end-to-end system, the bounding box would be generated using one of the many methods that have been developed for this purpose [52, 2, 8, 26, 10, 39]. Although we can no longer compute the angular difference between training and test images in this less controlled setting, this experiment allows us to determine how robust the different methods are to recognition against a real background and under occlusions. For this task, we use the same training set as before, *i.e.* training from just a single example. We also evaluate the performance as a function of the noise in the bounding box location and show that our method is robust to such variations.

4.1. Instance recognition from many examples

We first evaluate our method using the RGB-D object dataset [28], and we measure the performance when many training examples are available. This dataset consists of 300 objects of different types and textures, ranging from apples to cereal boxes. Given an image of one of these objects taken from a novel viewpoint, our task is to identify which of the 300 objects this image is taken from. We treat this task as a 300-class classification problem, and we are thus able to apply tools from machine learning to perform this task. Although the dataset consists of RGB images as well as depth, we only use the depth to obtain a segmentation of the target objects, both during training and at test time. In Section 4.4 we will explore the performance of the different methods when objects are placed in a real scene where a segmentation mask is not available.

We initially evaluate our method using the "leave sequence out" training set up of [28]. In this setup, we observe each object at a 30 degree and 60 degree elevation angle during training, and we observe the object at a 45 degree elevation angle at test time. During training, the object is placed on a turn-table, and we observe the object from many views spaced 6 to 9 degrees apart in azimuth.

The results for this setup when many training images are available can be seen in Table 1. We compare to the method of [28], which combines a number of visual features, including dense SIFT [38], texton histograms [36], and a color histogram. The methods that learn feature descriptors, such as [6] and [7] do significantly better, achieving accuracies between 90.4 and 92.1%. The results from [7] indicate that only small gains are achieved by adding depth

Method	% Accuracy
SIFT + Texton + Color Histogram [28]	60.7
Convolutional k-means descriptor [6]	90.4
HMP (RGB) [7]	92.1
Neural network (Ours)	**93.3**

Table 1. Training from many views: We first compare a neural network to previous methods for instance recognition when many views of each object instance are available during training. The neural network that we evaluate here is pre-trained only on ImageNet, with no multi-view pre-training.

information.

We evaluate the performance of a neural network pre-trained only with ImageNet (with no multi-view pre-training). Using such a network, we are able to outperform all of these previous methods, obtaining an accuracy of 93.3%. When many training examples are available for each object, we can achieve high performance without multi-view pre-training. However, in Section 4.2 we show that pre-training only on ImageNet has poor performance when the number of training examples per object is limited, thus motivating the use of multi-view pre-training for such cases.

4.2. Varying the number of training images

We note that the training setup from [28] is somewhat unrealistic. It is rare that objects are placed on a turn-table and that someone records images at so many different angles and elevations during training. In a typical product database, it is much more common to have only a few images taken of each object of interest. Further, a casual user will want to be able to recognize an object after taking only a few pictures during training.

We therefore create a new training setup to test the performance of these methods in a more realistic scenario. Each object is now viewed at training time at only a 30 degree elevation angle. We also vary the number of azimuthal angles for which an object is observed in training from 69 viewpoints down to just a single viewpoint. We evenly sample from the available training images for each object, starting from the first image. We can thus use this setup to determine how the performance of different methods are affected by the number of training examples.

Figure 2 shows the performance as we vary the number of views available during training. We compare the performance of the best methods of Section 4.1: HMP and a neural network pre-trained only on ImageNet (without multi-view pre-training).

As can be seen in this figure, the neural network saturates performance after about 10 training images. On the other hand, HMP [7] requires 30 training examples to saturate performance, and the result is still worse than that of the neural network. However, both methods perform poorly when only a single training example is available for each object. Because this is a situation that occurs often in practice, we would like to focus our attention on this scenario, which we call "one-shot learning for instance recognition." We will show that, when we have only one training example per object, we can improve the performance of a neural network by performing multi-view pre-training.

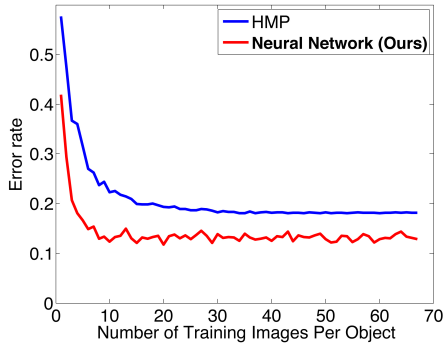

Figure 2. We observe the effect on performance as we vary the number of training examples for a neural network as well as for the HMP baseline. The neural network that we evaluate here is pre-trained only on ImageNet, with no multi-view pre-training. The y-axis in this plot is the error rate (not accuracy). These results are not directly comparable to those of Table 1 because in this setup we are training on only a 30 degree elevation angle, whereas for Table 1 we trained on both 30 and 60 degree elevation angles.

4.3. One-shot Learning for Instance Recognition

4.3.1 Baseline Methods

In the next experimental setup, we are given only a single training example of each object. At test time, we would like to recognize each object from novel viewpoints. We use the same test set as in Section 4.1, making this a strictly harder (though more realistic) training scenario. For all objects, we train on only one training image at a 30 degree elevation angle, and we test on many different azimuthal viewpoints at a 45 degree elevation angle.

The results for this setup are shown in Table 2. The keypoint-matching based methods perform poorly, ranging from 1.6% accuracy for BRISK [37] to 6.3% accuracy for SIFT [38]. More details about the baseline methods, as well as a further analysis of their performance, can be found in the appendix.

Machine learning methods perform significantly better on this task than the previous approaches. This is surprising because these methods are trained on just a single example per object, which is not common for machine learning approaches. HMP performs significantly better than the previous approaches when trained on just a single image, with an accuracy of 42.3%.

As can be seen in Figure 3, HMP performs well when the test example is viewed from a similar angle as the training example. However, the performance drops off quickly

Method	% Accuracy		
	Overall	Textured	Untextured
Random guessing	0.3	0.3	0.3
BRISK [37]	1.6	2.6	1.3
ORB [46]	1.9	3.5	1.3
SURF [3]	3.4	5.3	2.6
BOLD [50]	5.2	5.9	4.9
SIFT [38]	6.3	12.6	3.9
Line-2D [20]	5.5	0.3	7.4
Color Histogram Intersection [49]	12.4	23.3	8.2
HMP [7]	42.3	53.8	37.9
Neural Network (Ours)	59.2	63.2	57.6
Neural Network, MV + BG pre-train (Ours)	**63.9**	**73.8**	**60.0**

Table 2. One-Shot Instance Recognition: We compare our neural network approach to previous methods when only a single view of each object is available during training. The last row is our method with multi-view pre-training against random background images.

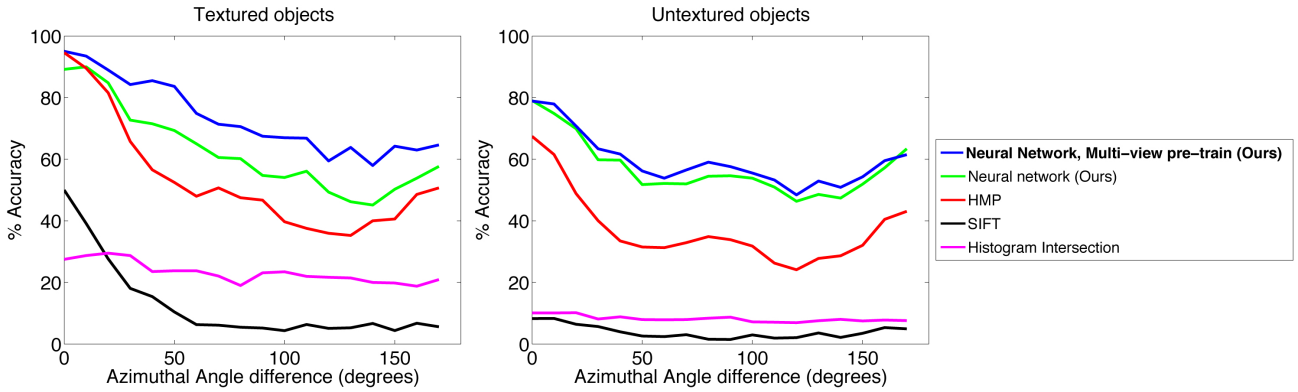

Figure 3. Average accuracy as a function of the azimuthal angle difference between test examples and the corresponding training example. Note that in all cases there is a 15 degree elevation difference between training and test images. The machine learning methods have a small increase in performance near 180 degrees due to the rotational symmetry of some of the objects.

as the angular difference between the training and test example increases. Note that, although we are varying the azimuthal angle difference from 0 to 180 degrees, all of the images have an additional 15 degree elevation angle difference between training and test. Given just a single training example, HMP is unable to find a good linear decision boundary that is viewpoint-invariant.

4.3.2 Neural Networks

We first evaluate the performance of a neural network that is pre-trained using ImageNet, as explained in section 4.1. After pre-training on ImageNet and fine-tuning on our dataset with just a single image per object, our network achieves an accuracy of 59.2%. Compared to the next-best method, this is an absolute improvement in accuracy of 16.9%, or a 29.3% drop in the number of errors. More neural network baselines can be found in the appendix.

We next experiment to see if we can gain an additional benefit from incorporating a separate multi-view dataset via multi-view pre-training. Note that the objects in the multi-view dataset are completely distinct from the 300 objects that we are trying to recognize. For this experiment, we use the multi-view BigBird dataset [48]. This dataset consists of 125 objects recorded from many different viewpoints, and to ensure that we have no overlap with the set of test objects, we remove the box of White Cheddar Cheez-it crackers, which also appears in the RGB-D object dataset [28]. We sample images from this dataset from 5 elevation angles and 20 azimuthal angles, for a total of 100 images per object. The multi-view dataset that we incorporate thus consists of a total of 12,400 images from 124 objects.

We evaluate the benefit of performing multi-view pre-training. In this setup, after pre-training our network on the 1.2 million images from ImageNet, we further pre-train our network to perform 124-class classification with the 124 objects from the multi-view dataset. Finally, we fine-tune the resulting network on the 300 objects from our single-view dataset, using just a single training example for each of the 300 objects. The details of our training procedure are described in Section 3.3.

Multi-view pre-training is especially impactful at im-

Figure 4. Examples that were classified correctly using multi-view pre-training but were incorrectly classified using a neural network pre-trained only on ImageNet. Left: Query image. Middle: Guess by neural network pre-trained only on ImageNet (incorrect). Right: Guess with multi-view pre-training (correct).

proving the recognition of textured objects. By pre-training with a multi-view dataset, we obtain a 10.6% absolute improvement (or a 28.8% reduction in errors) on recognizing textured objects, compared to the neural network pre-trained only on ImageNet. It is reasonable that multi-view pre-training gives a larger increase in performance on textured objects, since the appearance of textured objects changes more as a function of viewpoint compared to untextured objects. Thus, training our network to be invariant to rotations gives an especially large benefit for recognizing textured objects from novel viewpoints. At the same time, Table 2 indicates that multi-view pre-training improves our performance for untextured objects as well.

Note that the multi-view dataset contains only 1% as many images as were used in the original ImageNet pre-training step. It is surprising that, given only 1% more images, we obtain a 10.6% improvement on the recognition of textured objects. Figure 4 shows some examples of objects that our method was able to correctly recognize that were incorrectly recognized by a neural network pre-trained on ImageNet alone.

4.4. Objects in a Scene

In the previous set of experiments, we used test objects placed on a turntable so we could measure the rotational invariance of different methods in a controlled setting. However, for most applications we would want to be able to detect objects in a full scene, with a real background and occlusions. To measure whether our neural network with multi-view pre-training still gives the best performance in this more realistic setting, we used the RGB-D Scenes Dataset [28]. This dataset has per-frame bounding box annotations, which makes it suitable for our evaluation purposes. We crop the ground-truth bounding box from the scene and then classify the resulting image. The results can be found in Table 3. As can be seen, multi-view pre-training improves performance even for objects placed in an indoor setting with real background and occlusions.

Method	% Accuracy
Random guessing	0.3
BRISK [37]	9.4
ORB [46]	6.6
SURF [3]	10.8
BOLD [50]	7.4
SIFT [38]	12.9
Line-2D [20]	0.9
Color Hist Intersection [49]	9.2
HMP [7]	25.4
NN (Ours)	41.0
NN + MV + BG (Ours)	**44.1**

Table 3. One-Shot Instance Recognition in a Scene: We train each method from just a single example and test on cropped images from a full scene, with occlusions and real backgrounds. Note that the test set contains only a subset of the objects from Table 2, so the numbers are not directly comparable.

Notice that the single-view objects from our training set were recorded while placed on a turntable. To make our network robust to recognizing objects under novel backgrounds, the objects used for multi-view pre-training were synthetically placed against random scenes taken from the background category of the RGB-D scenes dataset [28], as explained in Section 3.2. In Table 4, we demonstrate the advantage of this multi-view pre-training against a random background. When the test images are depth-segmented, pre-training with a random background hurts performance slightly, with accuracy decreasing by 0.9%. However, when the test images are part of a real scene, pre-training with a random background increases robustness, improving accuracy by 2.6%. This demonstrates that pre-training on random backgrounds teaches our network to be robust to new backgrounds, even when the single-view objects being recognized are trained against a solid background.

We also analyzed the performance as a function of the noise in the bounding box location. To do this, for each bounding box we sampled a scaling factor s and a displacement Δx and Δy. These values are sampled from a distri-

Method	% Accuracy	
	Segmented	In Scene
NN	59.2	41.0
NN + MV (Ours)	**64.8**	41.5
NN + MV + BG (Ours)	63.9	**44.1**

Table 4. Comparison of different types of neural network pre-training. All methods are initially pre-trained on ImageNet. Top row: No multi-view pre-training. Middle: Multi-view pre-training with a black background. Bottom: Multi-view pre-training with random backgrounds. Test images are either depth-segmented (left) or taken from a real scene with background included (right).

bution that varies with a noise parameter n:

$$s \sim |\mathcal{N}(1, 0.025n)| \quad (1)$$

$$\Delta x \sim \mathcal{N}(0, 2n) \quad (2)$$

$$\Delta y \sim \mathcal{N}(0, 2n) \quad (3)$$

The test crop locations are then scaled by the scaling factor and shifted by Δx and Δy pixels. Examples of noisy images can be seen in Figure 5. Figure 6 shows the accuracy as a function of the noise parameter $n \in [0, 10]$; as seen, our method is robust to noise in the bounding box location and still significantly outperforms the baseline methods.

Figure 5. Left: Crops from a scene used to test robustness to background and occlusions. Right: The same crops with maximum noise added, to test robustness to bounding box noise.

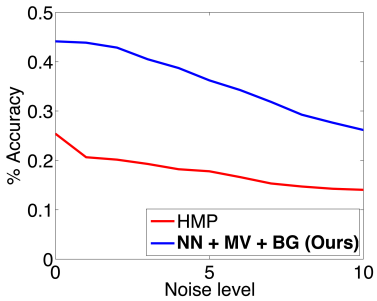

Figure 6. Instance recognition accuracy as a function of the bounding box noise parameter n.

4.5. Multiview Pre-training analysis

We can analyze which layers are benefiting most from multi-view pre-training. Recall that, for our experiments, we hold the convolutional layers fixed, as determined by cross-validation using a hold-out validation set (Section 3.3). Table 5 shows the effect of fixing different layers during multi-view pre-training, evaluated on the RGB-D objects dataset. If we hold the convolutional and both fully connected layers fixed, then we get the baseline performance (equivalent to not using multi-view pre-training). If we fine-tune just the fc7 layer during multi-view pre-training, then we get an improvement in performance of 1.7%. If we fine-tune both fc6 and fc7, then we get an additional improvement of 3% (for a total benefit of 4.7% over the baseline). Finally, if we also fine-tune the convolutional layers, then we get a further improvement of 1.2%. Thus, the biggest improvement seems to come from fine-tuning fc6. It seems that multi-view pre-training teaches the fully-connected layers the appropriate relationships between the convolutional features so that the network can be robust to viewpoint changes.

Method	% Accuracy
Baseline (no fine-tuning)	59.2
Fine-tuning fc7	60.9
Fine-tuning fc6 + fc7	63.9
Fine-tuning all	65.1

Table 5. Classification accuracy when fixing different numbers of layers during multi-view pre-training.

5. Conclusion

We are able to train a neural network to recognize objects from novel viewpoints given only a single training image of each object. By pre-training our network with multiple views of a separate set of objects, the network learns an increased robustness to viewpoint changes compared to pre-training only on class-level datasets. We show that neural networks with multi-view pre-training outperform previous state-of-the-art methods for instance recognition on both textured and untextured objects.

Thus, multi-view pre-training can make neural networks more robust to viewpoint changes. We also demonstrate that our multi-stage pre-training technique can be extended to learn other types of invariances, such as changes in background, by pre-training on objects with random backgrounds. We hope to extend this approach to pre-train on tracked objects from videos in the wild, allowing our network to learn to be more robust to occlusions, lighting changes, and many other types of changes that an object can undergo in the real world.

References

[1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, pages 329–344. Springer, 2014. 2

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *PAMI*, 34(11):2189–2202, 2012. 4

[3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417. Springer, 2006. 2, 6, 7

[4] S. Becker. Learning temporally persistent hierarchical representations. *NIPS*, pages 824–830, 1997. 2

[5] A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Advances in Neural Information Processing Systems*, pages 181–189, 2010. 2

[6] M. Blum, J. T. Springenberg, J. Wulfing, and M. Riedmiller. A learned feature descriptor for object recognition in rgb-d data. In *ICRA*, pages 1298–1303. IEEE, 2012. 4, 5

[7] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013. 2, 4, 5, 6, 7

[8] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, pages 3241–3248. IEEE, 2010. 4

[9] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod. Residual enhanced visual vector as a compact signature for mobile visual search. *Signal Processing*, 93(8):2316–2327, 2013. 2

[10] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, pages 3286–3293. IEEE, 2014. 4

[11] D. Damen, P. Bunnun, A. Calway, and W. W. Mayol-Cuevas. Real-time learning and detection of 3d texture-less objects: A scalable approach. In *BMVC*, pages 1–12, 2012. 2

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009. 2, 3

[13] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):594–611, 2006. 2

[14] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation by image exploration. In *Computer Vision-ECCV 2004*, pages 40–54. Springer, 2004. 2

[15] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[16] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. 2

[17] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014. 2

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587. IEEE, 2014. 1, 2, 3

[19] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng. Measuring invariances in deep networks. In *Advances in neural information processing systems*, pages 646–654, 2009. 2

[20] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of textureless objects. *PAMI*, 34(5):876–888, 2012. 2, 6, 7

[21] J. Hoffman, E. Rodner, J. Donahue, B. Kulis, and K. Saenko. Asymmetric and category invariant feature transformations for domain adaptation. *International Journal of Computer Vision*, 109(1-2):28–41, 2014. 2

[22] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and T. Darrell. One-shot adaptation of supervised deep convolutional models. *arXiv preprint arXiv:1312.6204*, 2013. 2

[23] E. Hsiao and M. Hebert. Occlusion reasoning for object detection under arbitrary viewpoint. In *CVPR*, pages 3146–3153. IEEE, 2012. 2

[24] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *PAMI*, 15(9):850–863, 1993. 2

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 3, 4

[26] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *ECCV*, pages 725–739. Springer, 2014. 4

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1, 2, 3

[28] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, pages 1817–1824. IEEE, 2011. 4, 5, 6, 7, 13, 14

[29] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, volume 172, page 2, 2011. 2

[30] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, pages 2526–2534, 2013. 2

[31] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using affine-invariant regions. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–319. IEEE, 2003. 2

[32] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178. IEEE, 2006. 2

[33] 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. 2

[34] C. Leistner, M. Godec, S. Schulter, A. Saffari, M. Werlberger, and H. Bischof. Improving classifiers with unlabeled weakly-related videos. In *CVPR*, pages 2753–2760. IEEE, 2011. 2

[35] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. *arXiv preprint arXiv:1411.5908*, 2014. 2

[36] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, 2001. 4

[37] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, pages 2548–2555. IEEE, 2011. 5, 6, 7

[38] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2, 4, 5, 6, 7, 10, 11

[39] S. Manen, M. Guillaumin, and L. V. Gool. Prime object proposals with randomized prim's algorithm. In *ICCV*, pages 2536–2543. IEEE, 2013. 4

[40] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM, 2009. 2

[41] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3d objects. *IJCV*, 73(3):263–284, 2007. 1, 2

[42] C. F. Olson and D. P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *Image Processing, IEEE Transactions on*, 6(1):103–113, 1997. 2

[43] X. Peng, B. Sun, K. Ali, and K. Saenko. Exploring invariances in deep convolutional neural networks using synthetic images. *arXiv preprint arXiv:1412.7122*, 2014. 2

[44] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014. 10, 12

[45] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3):231–259, 2006. 2

[46] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *ICCV*, pages 2564–2571. IEEE, 2011. 6, 7

[47] R. Salakhutdinov, J. Tenenbaum, and A. Torralba. One-shot learning with a hierarchical nonparametric bayesian model. 2010. 2

[48] A. Singh, J. Sha, K. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *ICRA*, 2014. 6

[49] M. J. Swain and D. H. Ballard. Color indexing. *IJCV*, 7(1):11–32, 1991. 6, 7, 10

[50] F. Tombari, A. Franchi, and L. Di. Bold features to detect texture-less objects. In *ICCV*, pages 1265–1272. IEEE, 2013. 6, 7

[51] T. Tommasi and B. Caputo. The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In *BMVC*, number LIDIAP-CONF-2009-049, 2009. 2

[52] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. 4

[53] L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002. 2

[54] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014. 3

[55] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014. 10, 12

A. Baseline Analysis

In our previous analysis, test examples are viewed from a 15 degree elevation increase with respect to the training image, and performance is shown as the azimuthal angle is varied. In Figure 7, we show the performance as the azimuthal angle is varied with no elevation difference between training and test images. When the azimuthal angle difference is small, most methods have a very good performance, especially for textured objects. However, for SIFT, Line-2D, and the color histogram, the performance quickly drops as the angle difference increases, showing a lack of robustness to changes in viewpoint. The neural network with multi-view pre-training is the most robust to changes in viewpoint, as demonstrated by this figure.

B. SIFT matching

In our previous analysis, we compared the results of our neural-network based approach to that of SIFT-matching. Here, we compare SIFT-matching with and without a RANSAC geometric verification step. We find that, for our dataset, we get lower overall accuracies when using SIFT with a geometric verification step, on both textured and untextured objects, as shown in Table 6. Because this result is rather unintuitive, we now provide a detailed analysis explaining why adding a RANSAC geometric verification step can sometimes hurt performance.

In both cases, we use a ratio test to filter matches, by comparing the distance from each SIFT match to that of its second-closest match, as was done in [38]. We search over thresholds for the ratio test and choose the best threshold, separately with and without the geometric verification step. We find that, when not using a geometric verification step, the highest accuracy is achieved by filtering with a ratio threshold of 0.6, whereas with a geometric verification step, the highest accuracy is achieved by filtering with a ratio threshold of 0.7. Note that a ratio threshold of 1 (between the closest match and the second-closest match) would be equivalent to not using a ratio test for filtering at all. Thus, for SIFT both with and without RANSAC geometric verification, filtering keypoints with a ratio test improves performance, as expected [38].

For some objects, adding a geometric verification step does improve SIFT performance. We show in Figure 8 a collection of examples for which SIFT with a geometric verification step is able to return the correct answer, whereas SIFT without geometric verification returns an incorrect answer. These examples include highly textured objects with significant image structure, for which SIFT is able to match a logo or design on the object across viewpoints.

However, in other cases, adding a geometric verification step can (surprisingly) hurt performance. See Figure 9 for examples which were classified correctly using SIFT keypoint-matching but were classified incorrectly when a geometric verification step is added. As can be seen in this figure, geometric verification can hurt performance when an object has repeated patterns. For example, on the soccer ball, keypoints on the black pentagons in the training image are matched to different pentagons in the test image in a way that is not geometrically consistent. When geometric verification is used, the soccer ball is not correctly matched because of the geometric inconsistency in these keypoint matches. A similar effect is seen with the other items in Figure 9, which have similar patterns that appear in different arrangements across the training and test images. For such examples, SIFT gives better performance without a geometric verification step. Still, the difference is relatively small, as shown in Table 6.

C. Color Histogram

For the color histogram method evaluated in the paper, we compute a histogram with 50 bins for hue and 60 bins for saturation. We then compare the training and test image histograms using the histogram intersection kernel [49]. We also compare histograms using a correlation, chi-squared distance, and Bhattacharyya distance, and found that the histogram intersection kernel performed the best.

D. Neural Network Baselines

We compare our multi-view pre-training procedure to a few alternative neural network training procedures, and show that our method significantly outperforms the alternatives. First, one might be concerned that, by fine-tuning the network on such a small dataset of 300 objects with just a single image per object, we would overfit our network. We therefore compare to a procedure in which we hold the entire network fixed and fine-tune only the final layer.

We also try using the output of different layers as an input to a linear SVM or a nearest-neighbor lookup, as has previously been done [44, 55]. For the nearest-neighbor lookup, because we have just 1 training example per class, we use kNN with k = 1. For the linear SVM, we use a one-vs-one SVM with voting [44]. Because each class has just a single training example, changing the C-value does not significantly affect performance; for our experiments we use C = 1.

The results are shown in Table 7. The three approaches that just use features from a fixed network pre-trained on

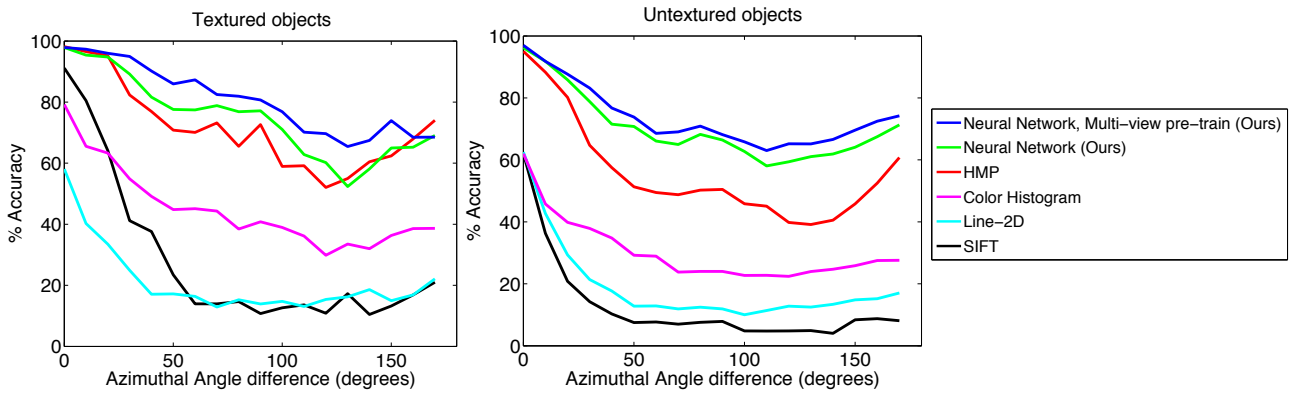

Figure 7. Average accuracy as a function of the azimuthal angle difference between test examples and the corresponding training example. For this experiment, there is no elevation difference between training and test images (only an azimuthal difference). Some methods have a small increase in performance near 180 degrees due to the rotational symmetry of some of the objects.

	% Accuracy		
Method	Overall	Textured	Untextured
SIFT [38]	6.3	12.6	3.9
SIFT with RANSAC geometric verification[38]	5.8	11.8	3.6

Table 6. Performance of SIFT keypoint-matching, with and without using a RANSAC geometric verification step. Overall, adding RANSAC hurts performance on this dataset, though it does help for some examples as well. See Appendix Section B for further discussion.

ImageNet perform poorly. The nearest-neighbor approach performs the worst, with an accuracy of 49% using fc7 features, and the SVM and softmax approaches perform only slightly better. Fine-tuning the entire network improves performance by 6.6% compared to training just the top layer. This is surprising because we are fine-tuning with just 300 training images, with only 1 image per class. As shown in the paper, a neural network with multi-view pre-training performs the best, improving performance by 4.7% overall and by 10.6% for textured objects.

E. Multiview Pre-training Analysis

E.1. Comparing Pre-training Strategies

Given that the original network was trained with 1 million images from ImageNet, it is surprising that we get a 4.7% overall improvement, and a 10.6% improvement on textured objects, after multi-view pre-training with just 1% more images. The number of images from the multi-view dataset is relatively small, and the objects in this dataset are distinct from the objects in our training and test sets. These results suggest that multi-view pre-training teaches our network to be robust to changes in viewpoint in a way that pre-training with images taken from separate object instances (*e.g.* ImageNet) does not.

However, one might still ask whether the improvement in performance is really due to the multi-view pre-training procedure or whether it is simply the result of our network seeing an additional 1% more images. Although this seems unlikely, we evaluate this hypothesis by comparing the performance of two training setups.

In both experiments, we start by pre-training our network on ImageNet. We then pre-train our network in two different ways. In the first multi-view pre-training approach, we pre-train our network using 20 views of each multi-view object, for a total of 2,480 training images ("Multi-view pre-train"). In the second pre-training approach, we have a separate class for each of 20 poses and each of 124 object instances, for a total of 2,480 classes ("Pose-class pre-train"). We use the same 2,480 training images in both cases. After pre-training our network using each method, we then fine-tune the network on the single-view object instances that we wish to recognize.

In summary, in the first setup, we have one class per object and we are training from multiple views of that object; in the second setup, we have one class per pose per object. However, in both cases, the training images being used are exactly the same; the only difference is the way in which we have defined the classes for multi-view pre-training. By comparing the performance of the network using these two different intermediate training procedures, we can determine whether the benefit of multi-view pre-training comes from learning viewpoint invariance due to our training procedure, or whether the benefit is simply the result of learning from more total images. Because the total number of images is the same in both cases, we can directly analyze the effect of our multi-view pre-training procedure.

The results are shown in Table 8. Using the standard neu-

Method	% Accuracy
kNN with pool5 [44]	34.3
kNN with fc6 [44]	47.6
kNN with fc7 [44]	49.0
SVM with pool5 [44, 55]	39.2
SVM with fc6 [44, 55]	50.7
SVM with fc7 [44, 55]	50.2
Neural Network, Fine-tune only top layer	52.6
Neural Network, Fine-tune all	59.2
Neural Network, MV + BG pre-train, Fine-tune all (Ours)	**63.9**

Table 7. Comparison of our method, in which we fine-tune the entire neural network (bottom), to approaches which hold all of the weights fixed except for the final layer.

Method	% Accuracy
Neural Network	59.2
Neural Network, Pose-class pre-train, 2,480 images	40.9
Neural Network, Multi-view pre-train, 2,480 images (Ours)	**62.3**

Table 8. We compare two versions of pre-training: one with a separate class per object and per pose (middle row) and one with just a separate class per object (bottom row). Both methods use the exact same images, but by using them in a different way we see a difference in the effect on performance. The top row is a neural network without any multi-view pre-training. All networks are initially pre-trained using ImageNet.

ral network (pre-trained on ImageNet), we get 59.2% accuracy. If we use multi-view pre-training on 2,480 images, we get 62.3% accuracy. On the other hand, if we pre-train a separate class for each pose of each object using the same 2,480 images, accuracy drops to 40.9%.

In the first case, our network learns to be viewpoint invariant and shows improved performance on the final single-view dataset. In the second case, our network learns to distinguish between poses, and shows a significant drop in performance on the final single-view dataset. In the second case, even though the network was pre-trained on the same set of images, they were not used in a way that allowed the network to learn to be viewpoint invariant. Thus, the improvement of multi-view pre-training is not just that the network has "seen" more images, but by pre-training to classify objects from different viewpoints, the network learns to be robust to changes in viewpoint.

E.2. Comparing to Data Augmentation

We propose to use multi-view pre-training as a way to teach our network to be robust to changes in viewpoint. However, another strategy to train a network to be robust to changes in viewpoint is to simulate changes in viewpoint via data augmentation. From just a single image, this is difficult for non-planar objects. Still, we can simulate perspective warps, which will be a correct transformation for fronto-parallel planar objects, and might still be a useful transformation for non-planar objects.

The results are shown in Table 9. Using perspective warps during data augmentation does increase the robustness of the network to new viewpoints. To the best of our knowledge, this is a novel type of data augmentation that has not been explored previously.

Using multi-view pre-training, our method gives an even bigger improvement. However, besides the effect on performance, there are a number of other reasons to perform multi-view pre-training rather than (or in addition to) performing data augmentation. First, our results are obtained from multi-view pre-training with just 124 objects. If more objects are used for this stage, then our performance might be expected to improve even further. Additionally, we demonstrate that multi-view pre-training can teach our network to be robust to both viewpoint changes as well as changes in background. By tracking objects that undergo other types of changes, our network can use multi-stage pre-training to learn to be robust to even more types of changes, such as changes in lighting, occlusions, or deformations. Multi-stage pre-training is thus a general technique that can be used to learn different invariances by observing how objects change their appearance over time.

Finally, multi-view pre-training leads to a faster convergence, as shown in Figure 10. Because the method has already learned to be robust to changes in viewpoint, it can quickly learn to recognize new objects from novel viewpoints, without the need for extensive data augmentation. The figure shows that the slowest convergence is achieved using data augmentation with perspective warps. Faster convergence is achieved if multi-view pre-training was used so that the network has already learned to be robust to changes in viewpoint.

Method	% Accuracy
Neural Network	59.2
Neural Network, Perspective Augmentation	62.9
Neural Network, MV + BG pre-train (Ours)	**63.9**

Table 9. We compare multi-view pre-training (bottom) to data augmentation by perspective warping (middle). The top row is a neural network without any multi-view pre-training or perspective warping. All networks are initially pre-trained using ImageNet.

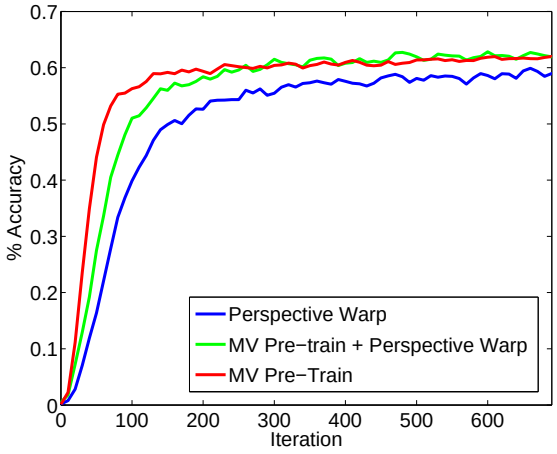

Figure 10. Convergence rates for three different approaches to learning robustness to changes in viewpoint. Multi-view pre-training allows our network to quickly learn to recognize new objects.

F. Error Analysis

It is instructive to look at what kind of errors our network still makes after multi-view pre-training. Although multi-view pre-training teaches our network to be robust to changes in viewpoint, there are other types of changes that our network is still not robust to. For example, Figure 11 shows that our network is not robust to subtle color differences between objects, or changes in color due to the scene lighting. Some errors are caused by the depth-segmentation that is used to pre-process the images (Figure 12). We use the pre-processed images from Lai, et al [28]. Other errors are caused by the limited viewpoint available during training, or by poor bounding-box localization of objects in a scene, as shown in Figure 13.

Figure 11. Examples that were classified incorrectly even after performing multi-view pre-training. These examples demonstrate that our network is not robust to subtle color differences (top two rows) or changes in lighting (bottom 4 rows). Left: query image. Middle: guess by neural network with multi-view pre-training. Right: the correct match.

G. Experimental Details

In many of our experiments, we compute performance separately for textured vs untextured objects. These divisions were chosen by the authors, and we considered the following object categories to be textured: cereal box, food bag, food box, food can, food cup, food jar, instant noodles, shampoo, soda can, and water bottle (see Figure 14). The remaining categories were considered to be untextured: apple, ball, banana, bell pepper, binder, bowl, calculator, camera, cap, cell phone, coffee mug, comb, dry battery, flashlight, garlic, glue stick, greens, hand towel, keyboard, kleenex, lemon, light bulb, lime, marker, mushroom, notebook, onion, orange, peach, pear, pitcher, plate, pliers, potato, rubber eraser, scissors, sponge, stapler, tomato, toothbrush, and toothpaste (see Figure 15). Some objects could have been placed into either category, but the main purpose is to observe general trends in recognizing textured vs untextured objects, so individual categorization choices are less important.

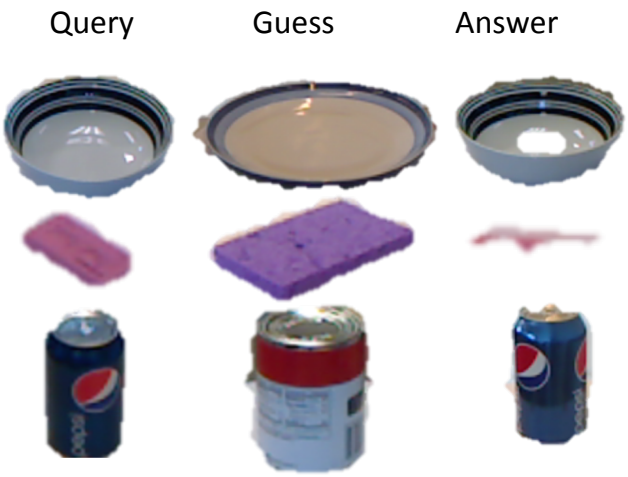

Figure 12. Examples that were classified incorrectly even after performing multi-view pre-training. These errors seem to be caused by the depth-segmentation that is used to pre-process the images [28]. Left: query image. Middle: guess by neural network with multi-view pre-training. Right: the correct match.

Figure 13. Examples that were classified incorrectly even after performing multi-view pre-training. Some errors are caused by the limited viewpoint available during training (top) or poor bounding box localization for objects in a scene (bottom). Left: query image. Middle: guess by neural network with multi-view pre-training. Right: the correct match.

Figure 14. Examples of textured objects from the RGB-D object dataset.

Figure 15. Examples of untextured objects from the RGB-D object dataset.

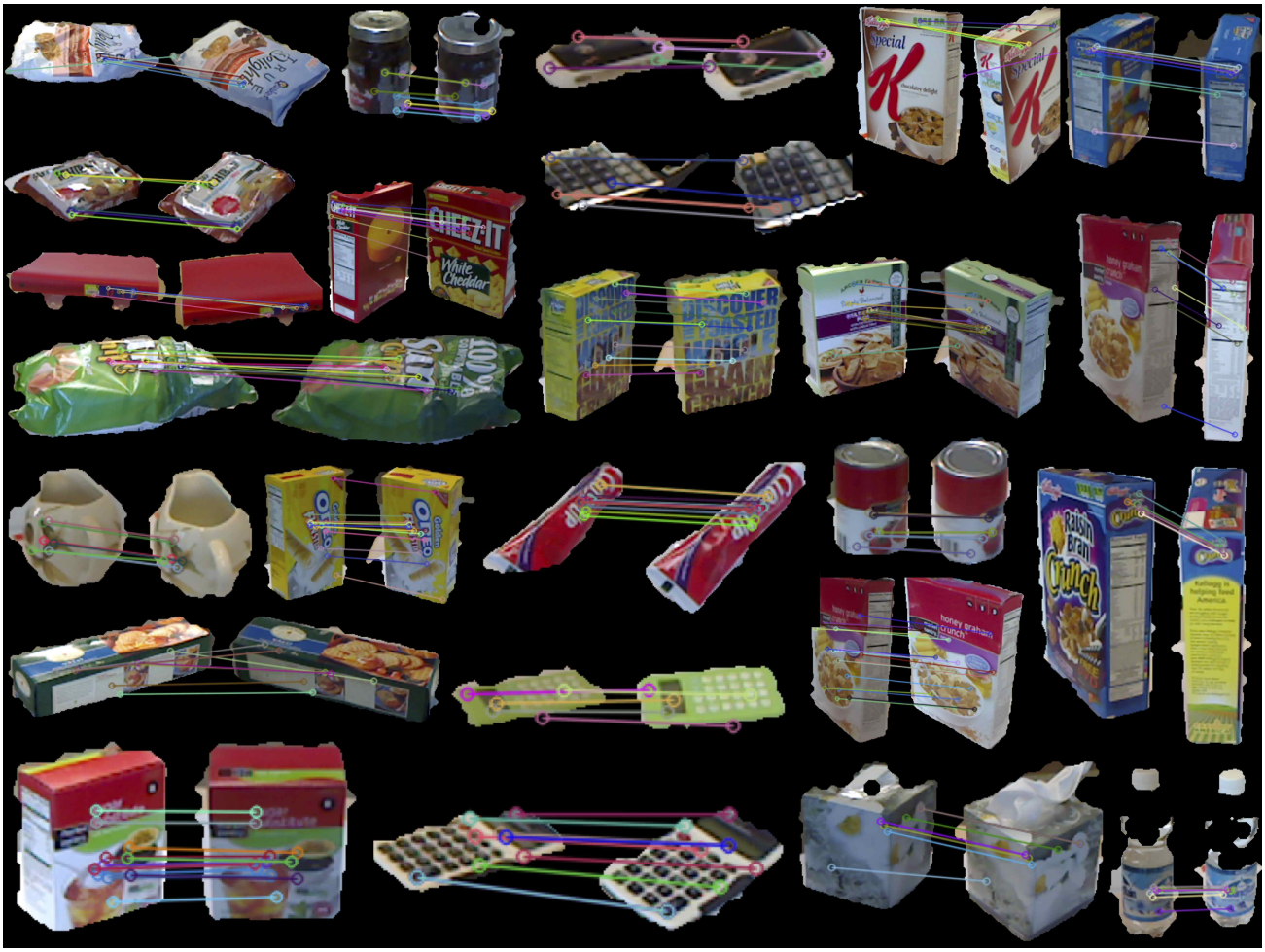

Figure 8. Examples that were classified correctly using SIFT with a RANSAC geometric verification step but were classified incorrectly when SIFT was used without a geometric verification step. On some objects, adding a geometric verification step can improve performance, as expected.

Figure 9. Examples that were classified correctly when SIFT was used without a geometric verification step but were classified incorrectly when adding a RANSAC geometric verification step. On some objects, adding a geometric verification step can actually hurt performance. Note that the boxes on the bottom left and bottom right are attempts to match the front to the back of the packaging, which have similar elements placed in a different arrangement.