Measuring Occlusion Invariances in Deep Networks

Kyunghee Kim
Oracle Corporation
kakooyang@gmail.com

Abstract

In computer vision, finding robust features invariant to many artifacts such as translation, reflection, occlusion, and view angles is important to improve the performance of the classification. Goodfellow et al.[1] measured invariances to translation and rotation of the features learned from stacked autoencoder networks[2] and convolutional deep belief networks[3]. This project measures invariances to the occlusions of those deep networks with two object categories. Car image data and computer image data from the image dataset LabelMe[4] were used to train and test deep autoencoder networks[5] and convolutional deep belief networks[3]. The result with car image data show that in the test using deep autoencoder networks the occlusion invariance score achieved better score in layer 2 than in layer 1. However, layer 3 was measured with the lowest score among three layers. Convolutional deep belief networks show higher invariance score in layer 1 than in layer 2. The test with computer image data learned more invariant features in layer 2 than in layer 1. Even if the most of the result does not comply with the general characteristics of deep networks that deeper layers learn more invariant features than shallower layers, this result suggests that we need to further examine the properties of deep networks more specifically with regard to translation, rotation, occlusion, view angles, illumination and so on.

1. Introduction

Image data in computer vision might have various artifacts such as illumination, rotation, translation, occlusion, etc. Human eyes can recognize objects even if the appearance of the object is affected by those variances. To achieve the high performance of computer vision in object recognition problem, the features extracted from the objects also have to be invariant to the transformation or deformation so that the classifier can recognize objects despite the presence of variances in the input.

Deep architectures consist of several layers and the hierarchical architectures in deep learning make it possible for upper layers to learn more invariant features than lower layers[2,3,5,6]. Goodfellow et al. presented formulas to calculate the invariance score in each layer in deep architectures and showed that stacked autoencoder networks and convolutional deep belief networks learn more robust feature representations when rotation was applied to the input[1]. This project presents a formula to measure invariances in deep networks when the object in the input image is occluded and measures the invariance scores in deep autoencoder networks and convolutional deep belief networks.

2. Related Work

Hinton et al. made the deep belief networks with a restricted Boltzmann Machine in each layer[5]. They showed that lower layers in the network learn low-level features and deeper layers in the networks learn more abstract features. Bengio et al. presented a deep network with an autoencoder neural network in each

layer[2]. Lee at al. built a convolutional deep belief network using probabilistic max-pooling and showed that their architecture learns useful object features[3].

Goodfellow et al. measured invariances in deep networks on synthetic images and natural video data[1]. They tested two transformations i.e., translation and rotation. While invariance scores to the rotation were measured higher in deeper layers in both stacked autoencoder networks and convolutional deep networks, invariance score to the translation in stacked autoencoder networks was the lowest in the deepest layer.

3. Occlusion and Data Collection

LabelMe[4] dataset contains good examples of occlusions of cars. Some of these examples are illustrated in Figure 1.

(a) (b) (c)

Figure 1. This figure shows three examples found in LabelMe[2] dataset. (a) shows one object(car) with its full body. (b) shows one object(car) occluded by a bar in front of it. (c) shows two cars; One is occluded by a red car in front of it and the other(red car) is occluded since the original picture did not take its full body.

Figure 2 shows three examples of training dataset and three examples of test dataset. Figure 2 (a)-(c) are training examples and Figure 2 (d)-(f) are test examples. Each image was cropped from the images in LabelMe[4] dataset, which contain a car as a object in them and each cropped image was rescaled to 64 by 64 pixels and also grey scaled to a black and white image. The training data and test data for computer are attached in Appendix.

(a) (b) (c) (d) (e) (f)

Figure 2. Training data and test data of cars used in this experiment

4. Deep Network Architectures

4.1 Deep Autoencoder Networks

A deep belief network with a restricted Boltzmann machine in each layer introduced by Hinton et al.[5] was used for this experiment. In each layer, the activation of each unit, h_i, $i = 1,..., m$, to an input, $x \in R^n$, is computed as

$$h(x) = sigmoid(Wx + b),$$

where $h(x)$ is the unit activations, W is a weight matrix, b is a bias in hidden layer and *sigmoid* is a sigmoid function. The networks in this experiment consists of three layers and were trained in a greedy layer-wise way as presented in [2]. The first layers receives 25 by 25 patch of an image as an input. The training set size of car images is 200 and test set size of car images is 125. The training set size of computer images is 150 and the test set size of computer images is 100. The batch size is 25 in both cases.

4.2 Convolutional Deep Belief Network

CDBN[3] consists of two layers in this experiment. Each layer contains convolutional units and max-pooling units. The receptive field size of each convolutional unit is 10 by 10 pixels. Each max-pooling unit has the receptive field size of 11 by 11 pixels in the first layer and 31 by 31 pixels in the second layer. The size of the receptive field in each max-pooling unit is determined by implementing max-like operation over four neighboring convolution units. The training set size of car images is 203 and test set size of car images is 143. The training set size of computer images is 150 and the test set size of computer images is 100.

5. Occlusion Invariance Measure

As a feature detector, a hidden unit has to respond strongly if there exists a feature that it represents in the input. And if the feature it represents does not exist, the hidden unit must not respond to the input. To be invariant to the input, the hidden unit should respond to the input even if there has been applied some transformations or deformations to the input such as translations, reflections, rotations, and occlusions. Therefore, if the hidden unit is invariant to occlusions it has to respond to the input strongly even if some part of the object is hidden or not seen for some reason. For example, an object might be hidden by some other objects or the original image was taken with only the part of the object as shown in Figure 1 already in part 3.

To ensure that the units are selective, *Global firing rate*, *G(i),* is defined to be the proportion of the units that respond to the features present in the inputs. In this experiment the *Global firing rate* was set to be 0.01. The *Global firing rate* is also used to compute the threshold, t_i, which determines whether the unit is *'on'* or *'off'*.

To measure the robustness of the units, *Local firing rate*, *L(i),* is defined as the proportion of the units that fire with the input. The unit is determined to be *'on'* when it is over the threshold, and *'off'*, otherwise. To express it with an indicator function, it is defined as $f_i(x) = 1\{hi(x) > t_i\}$. *O(x)* is a set of stimuli that deforms the input with occlusion. *Z*, similarly as defined in [1], is a set of inputs that near maximally activate the units. Finally, the Local firing rate is defined as

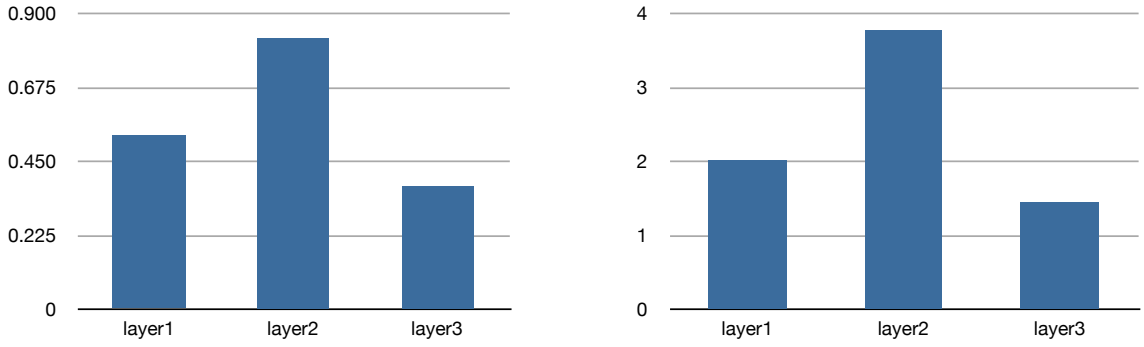
$$L(i) = \frac{1}{|Z|}\sum_{z\in Z}\frac{1}{|O(z)|}\sum_{x\in O(z)} f_i(x),$$

Invariance score for a unit $h_i(x)$ is defined *Local firing rate* over *Global firing rate* i.e., *S(i) = L(i)/G(i)*.

6. Results

6.1. Deep autoencoder Networks

Figure 3 shows the occlusion invariance test result using deep autoencoder networks with car image data. The second layer learned more invariant features than the first layer. However, the third layer learned the least invariant features.



(a) Average invariance score of all units

(b) Average invariance score of top 20% units

Figure 3. Occlusion invariance test result in deep autoencoder networks with car image data. For both all units and top 20% units, $p < .0001$.

Figure 4 shows the occlusion invariance score in deep autoencoder networks with computer image data. The first layer learned the most invariant features and the second layer learned the least invariant features.

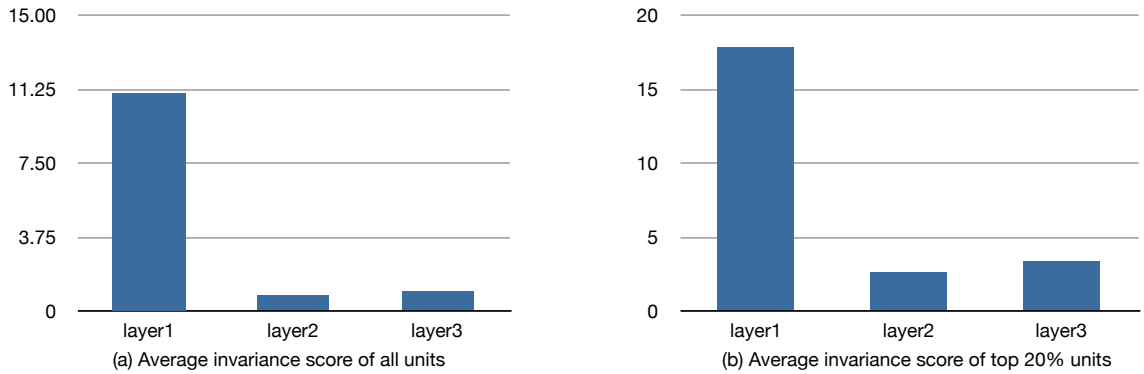


Figure 4. Occlusion invariance test result in deep autoencoder networks with computer image data. For both all units and top 20% units, $p < .0001$.

6.2 Convolutional Deep Belief Networks

Figure 5 shows the occlusion invariance test result in CDBN with car image data in Figure 5 (a) and with computer image data in Figure 5 (b). In the test with computer data the second layer learned more invariant features than the first layer, whereas in the test with car data the first layer learned more invariant features.

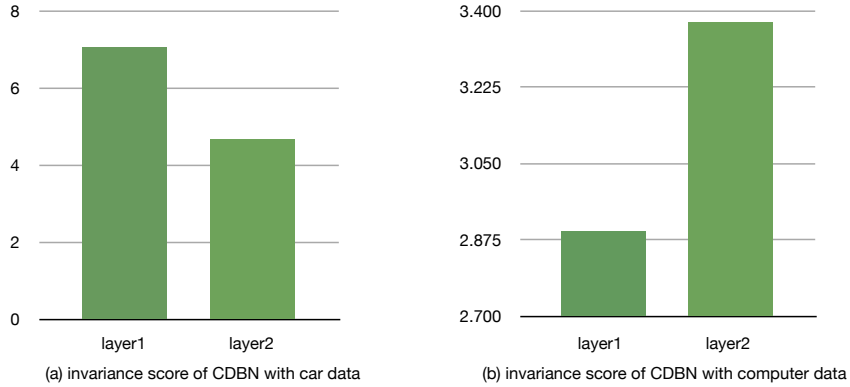


Figure 5. Occlusion invariance test result in CDBN with car image data (a) and computer image data (b)

7. Discussion and conclusion

This experiment measured the invariance scores in deep autoencoder networks and convolutional deep belief networks when occlusion was present in the image data. Two object categories were tested, i.e., car images and computer images. As a result, only the test using CDBN with computer images showed the result that agrees with the property of the deep networks generally thought that deeper layers learn more invariant features then shallower layers. Other tests showed the result different from this generally believed property of deep networks as illustrated in part 6. Results above. Meanwhile, in the test with stacked autoencoder in Goodfellow et al.[1]'s work, the invariance score to the translation decreased as the layer went higher, which also did not comply with the characteristics of deep networks. However, in their work, the invariance test with rotation and the invariance test using CDBN with translation showed the result that follows the general characteristics of deep networks. To summarize all these results in previous work and in this experiment, we need to further examine properties of deep networks with respect to various transformations and deformations in different architectures. The suit of formulas suggested in this work and previous work presented by Goodfellow et al. provides a way to calculate to the invariances in deep networks. Lastly, to mention the weakness of the dataset in this experiment, the data includes not only the

effect of occlusions but also other artifacts such as illuminations or different view angles. For example, to compare the two images in Figure 2 (a) and (b), their view angles are slightly different from the perspective of the viewer who took these photos. Therefore, as a future work, we need to independently test each artifact and we need to test more diverse object categories whereas this work tested two object categories.

References

[1] I. J. Goodfellow, Q. V. Le, A. M. Saxe, H. Lee, A. Y. Ng. Measuring Invariances in Deep Networks. NIPS, 2009.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. NIPS, 2007.

[3] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical represenations. ICML, 2009.

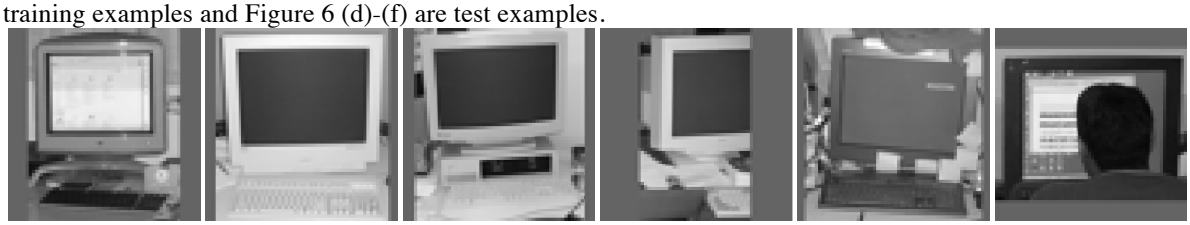
[4] B.C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *MIT AI Lab Memo*, 2005.

[5] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief networks for scalable unsupervised learning of hierarchical representations. ICML, 2009.

[6] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. ICML, 2007.

Appendix.

Figure 6 shows three examples of training dataset and three examples of test dataset. Figure 6 (a)-(c) are training examples and Figure 6 (d)-(f) are test examples.



(a) (b) (c) (d) (e) (f)

Figure 6. Training data and test data of computers used in this experiment