Anomaly Detection in Image Data Sets using Deep Learning

Ali Radha, Kumar Saketh,

Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 48109

Abstract

In this project, we are interested in designing an algorithm for automatically detecting anomalies in image data sets. To get around hand-engineering features, we plan to exploit current work on deep learning to automatically extract features in an unsupervised manner. In particular, we plan to learn deep networks (in an unsupervised manner) from the unlabeled image data set of interest, and subsequently use the learned network in two different ways to detect anomalies: (i) detect anomalous images using the features learned by the network, and (ii) detect anomalous images based on the pattern generated by the active nodes corresponding to each image.

Introduction

Anomaly detection is the task of identifying entities in a data set which differ from the nominal majority. Anomaly detection has been studied extensively for structured multivariate data where each entity is represented by a set of *d* features. Recently, there has been some work on solving anomaly detection for unstructured data such as time-series, network graphs and images . However, most of these methods rely on extracting hand-engineered features from the unstructured data and subsequently applying anomaly detection methods for structured data.

With the advent of deep learning, feature representations can now be learned automatically from large unlabeled collections of images. It has been shown that the features inherently learned by these deep architectures have been successful in several tasks in computer vision including classification and segmentation. We believe that these deep architectures can also be exploited to detect anomalous images in large image data sets.

We are keen to build off of this work and use the features learned from deep neural networks to detect anomalous images. If we are successful, our proposed idea of using deep learning to detect anomalous images will have the advantage over traditional hand-engineering based methods in that (i) the algorithm will generalize to a wide variety of image data sets, and (ii) the performance of our algorithm, as is the case in classification tasks, could potentially be better that the performance of hand-engineered systems. For these reasons, we are excited to work on this problem.

Implementation Progress

Thus far, we have worked on identifying anomalies in the MNIST data set. To identify anomalies, we have implemented the current state-of-the-art method based on reconstruction errors. We have also implemented our proposed approach of building an unsupervised auto-encoder on the MNIST images, and then using the features learned by the auto encoder as input to the *k*-nearest neighbor (kNN) anomaly detection algorithm and the Isolation Forest anomaly detection algorithm in order to detect anomalies.

We have successfully implemented a single hidden layer auto-encoder (Figure 1) in Matlab. This auto encoder learns features by minimizing reconstruction error between the input, and the output composed of the learned features. The input and output nodes of this network are equal to the number of pixels in each MNIST image. After training the auto-encoder network, we used the network to represent each MNIST image as a 50-dimensional vector corresponding to the 50 nodes in the hidden layer.

We also implemented the Isolation Forest (iForest) algorithm in Python. The iForest algorithm involves construction of isolation trees, which recursively partition the data by randomly splitting the data along a randomly chosen axis. The number of cuts needed to isolate a point is then used as an indicator of the anomalousness of the point - the more easy it is to isolate a point (i.e., smaller isolation depth in the tree), the more anomalous the point is. We computed a total of 2000 isolation trees and used the average isolation depth of each point as the measure of anomalousness.

Comparison of Results

In the MNIST dataset, because we do not have labels on which points are normal and which are anomalous, we are currently restricted to qualitatively comparing the performance of these algorithms. In Figure 1, we display the top 20 images for each of the following categories: (i) images with lowest reconstruction error, (ii) images with highest reconstruction error, (iii) images with smallest isolation depth, and (iv) images with highest kNN distance.

Based on these results, we make the following observations: (i) the results of the kNN method seem to be qualitatively better than both the reconstruction method and the iForest method - in particular, the later two algorithms seems to be picking several '1' digits that look fairly normal to us; (ii) it also appears that the reconstruction error methods picks digits which are anomalous because of their orientation; (iii) the kNN methods picks digits which are anomalous because certain strokes are exaggerated in length (for e.g., see 4's in the kNN image set).

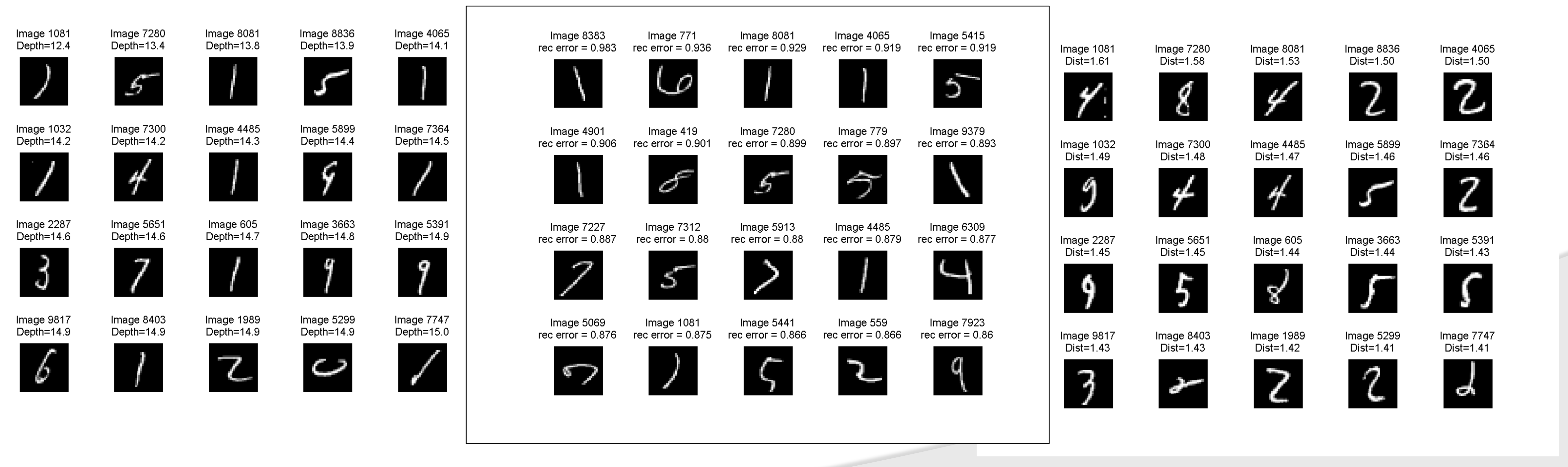

Figure 1: Four sets of MNIST images. From left to right: Anomalies wrt. reconstruction error, Isolation forest, kNN.

Future Work

We would like to study the current results in further depth - in particular, to understand why the performance of iForest is relatively worse compared to kNN though both are operating on the same feature spaces. We would also like to understand if there is a fundamental difference in the type of anomalies being identified by the reconstruction method and the feature based methods.

Next, we would like to implement a method to quantitatively compare the performance of our algorithms. We plan to do this by constructing anomaly detection data sets from classification data sets, and then using the Area under the ROC curve as a quantitative metric.

We plan to run our algorithms on noisy image data sets like ImageNet in order to see if our proposed schemes are more robust in the presence of background noise relative to the reconstruction based method.

Finally, we would like to explore alternative neural network architectures including multi-layer autoencoders and deep belief nets.