Comparing supervised and unsupervised learning in American Sign Language Recognition

Archana Anand
Department of Computer Science
University of Colorado Boulder
archana.anand@colorado.edu

Shipra Behera
Department of Computer Science
University of Colorado Boulder
shipra.behera@colorado.edu

Yuchen Zhang
Department of Computer Science
University of Colorado Boulder
yuzh6480@colorado.edu

Abstract

The problem we are investigating is the comparison of supervised and unsupervised learning performance for American Sign language recognition. Many researchers have investigated adaptive technologies which enable people to perform tasks with great efficiency, hence increasing their quality of life. There has been a lot of ongoing work, particularly involving neural networks[1], to address this problem. We will investigate different supervised and unsupervised techniques for the automatic recognition of signs and evaluate their accuracies over the 24-signs.

1 Introduction

Sign language is a hand gesture language which the hearing impaired use to express themselves. American Sign Language (ASL) significantly facilitates communication for most hearing impaired individuals. Fewer than half million people in the United States use ASL[9], which significantly restricts the number of individuals they can communicate with. Sign language recognition has been an interesting computer vision problem for decades. Developing a better understanding of performance given by unsupervised learners compared to that given by supervised learners is critical, considering that in real world scenario, humans learn mostly in an unsupervised manner. Through this project, we want to explore and compare different machine learning algorithms - in both supervised and unsupervised setting.

2 Related Work

ASL recognition is not a new computer vision problem.Over the past two decades, researchers have used classifiers from a variety of categories that we can group roughly into linear classifiers, neural networks and Bayesian networks. [1] uses various supervised learning for sign language classification. Their Neural Network gives accuracy of 96% when used with 10 classes. Our highest accuracy with Neural Network is 94.33%with all 24 classes. Our SVM accuracy of 86.67%is better compared to one in the paper which is 62.3%. Use of correct feature extraction can be the reason for such high accuracy. [2] uses Convolution Neural Network with accuracy of 82%. [3] shows similar work of using different supervised learning with feature extraction. So far we have not seen any paper working on unsupervised learning methods. This experiment is mainly to compare supervised and unsupervised learning accuracy using the technique discussed in section 4 of this paper.

3 Dataset

We are using the American Sign Language MNIST dataset on Kaggle[4]. The dataset format is very similar to the classic MNIST handwritten digit dataset. Each training and test data point has a label 0-24(except 9) as a one-to-one mapping for each alphabetic letter A-Z (except 'J' and 'Z' since both of them are not static signs and required multi frame processing for their identification). The training data (27,455 samples) and test data (7172 samples) have a feature vector of dimension 784 (i.e a 28x28 pixel image) and grayscale values between 0-255.

4 Proposed Work

Our work involves comparing different supervised and unsupervised learning algorithm before and after feature extraction technique. We are comparing the accuracy of supervised and unsupervised algorithm. The tricky part here is to get the accuracy in an unsupervised setting. All of our unsupervised learning algorithms are clustering algorithms. We use different clustering methods to form the clusters for a given training set. Assuming that our test set has its true label, we get the predicted labels by our cluster on this test set. This test set is then used to map each cluster label to the true label i.e if majority of the test data classified inside a cluster is labeled 1, we map the cluster to label 1. The number of misclassified data points then becomes our test error. We would discuss these algorithms and results in details below

5 Feature Extraction/Dimensionality Reduction

Good segmentation process can lead to perfect feature extraction process which plays an important role in successful recognition process. There are many interesting points in an object which can be extracted to provide a feature descriptor of the object.

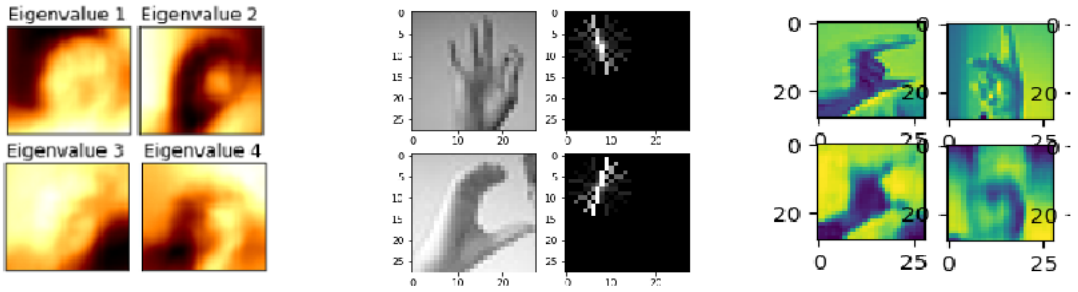

Figure 1: PCA, HOG and Autoencoder

5.1 PCA

Principal Component Analysis[5] is a feature extraction technique which eliminates least discriminative features i.e features having low variance. [Fig 1] shows the decreasing eigenvalue of the image.The experiment and graph below(Fig [2-G] below) shows Single Variable Variance(with x axis being dimension and y axis being variance of the component along a particular dimension). PCA is a good feature extraction technique with those algorithms where high dimensional data leads to Curse Of Dimensionality, for example DBSCAN(Fig [2-H] below). DBSCAN algorithm gives 0 cluster with high dimensional data but after applying PCA, we can see distinct clusters getting formed. PCA may destroy important information which otherwise can be used by different algorithms not affected by high dimensionality. This can be one reason why some of the algorithms e.g SVM and Naive Bayes, give poor accuracy compared to its original accuracy.

5.2 HOG

Histogram of Oriented Gradients(HOG)[6] counts the occurrences of gradient orientation in localized portions of an image.The local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. HOG features are calculated by taking orientation histograms of edge intensity in local region. The implementation of these descriptors can be achieved by dividing the image into small connected regions, called cells, and for each cell compiling a histogram of gradient directions or edge orientations for the pixels within the cell. The combination of these histograms then represents the descriptor. We can see that HOG is a good descriptor for object detection, and good performance can be achieved with Random Forest and linear SVM. One can expect even better performance with kernel SVM, if the computational complexity is not considered. DBSCAN relies on the density of data points so sparse data sets may not work well and lose efficiency on higher dimensional data sets such as this one.

5.3 Convolutional Autoencoder

Convolutional autoencoder, a type of artificial neural network, is one efficient data coding technique, which is widely used in computer vision, especially used in dimensionality reduction, image noise reduction. The model learns a representation for a set of data, in lower dimension than the original, and can recover the data by a decoder net. The last graph in Figure 1 displays the raw data (first row) and recovered data (second row) from encoded ones.

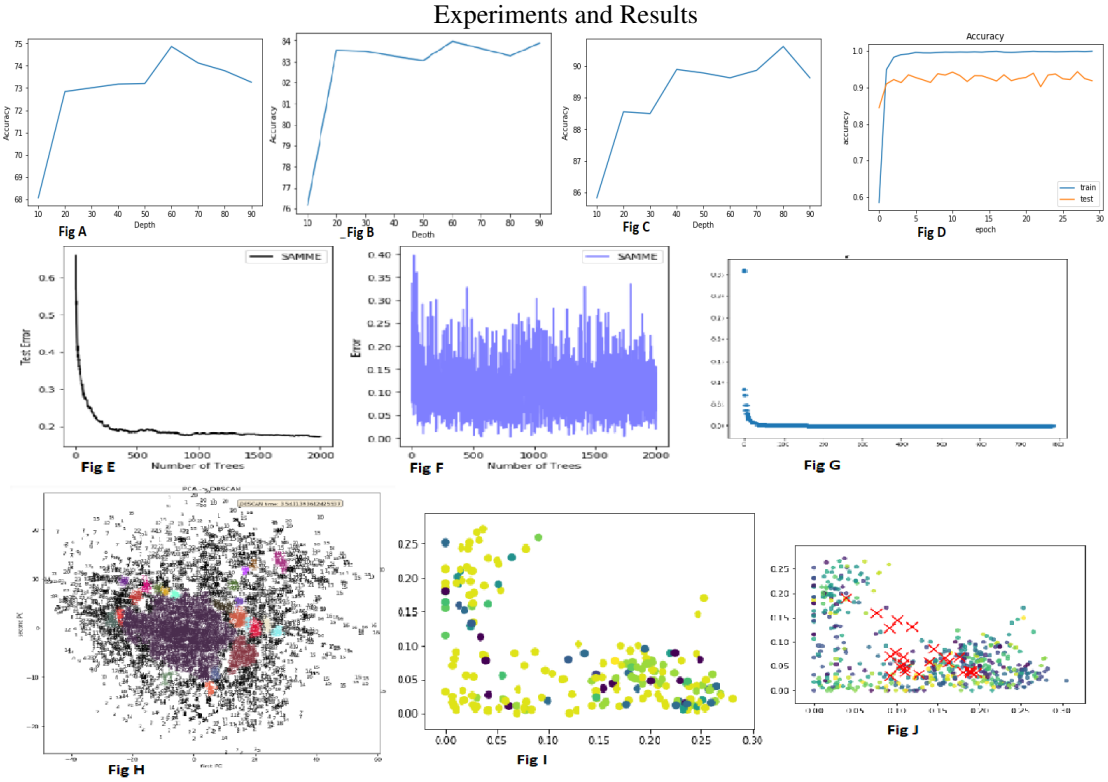

Figure 2: Fig A-C are the plots for test accuracies of Random Forest with feature descriptors Autoencoder, PCA and HOG respectively. We can see that HOG achieves high accuracy quickly with less depth as compared to others. Fig D is the plot for training and test accuracies on neural network. Fig E and F show the variation of test error over number of trees for AdaBoost classifier. Test error rate decreases as we increase the number of trees. Fig G shows the single variable variance for PCA. Fig H, I and J visualize the cluster formation for DBSCAN, GMM and Kmeans respectively over HOG descriptors.

Table 1: Comparison of accuracies over different classifiers

Classifier	Accuracy
Neural Network	94.33 %
Random Forest	82.31 %
SVM	60.76 %
AdaBoost	51.05 %
Naive Bayes	38.98 %
Kmean	24.05 %
GMM	16.66 %

Table 2: Comparison of accuracies over different feature descriptors and classifiers

Feature Descriptor	Parameters	Classifier	Accuracy
HOG	Depth = 40, Estimator = 400	Random Forest	90.61 %
HOG	1 vs Rest	SVM	86.67 %
HOG		Naive Bayes	72.60 %
HOG	24 clusters	Kmean	50.55 %
HOG	24 clusters	GMM	24.67 %
PCA(40 components)	Depth = 40, Estimator = 400	Random Forest	83.40 %
PCA	Number of weak classifiers =2000, Depth of tree = 20	AdaBoost Multiclass	82.9 %
PCA	1 vs Rest	SVM	57.4 %
PCA		Naive Bayes	16.7 %
PCA		DBSCAN	12.50%
Autoencoder	Net Structure	Random Forest	74.86 %
Autoencoder	Net Structure	SVM	63.31 %
Autoencoder	Net Structure	AdaBoost	36.60 %
Autoencoder	Net Structure	Naive Bayes	34.94 %
Autoencoder	Net Structure	Kmeans	24.12 %

In Table[1], we first establish a baseline of accuracies over different classifiers without using any feature descriptors. In Table[2], we show how using different feature descriptors can affect the accuracy of prediction on test data. For supervised learning we are using Neural Network[8], SVM, Random Forest, Naive Bayes, AdaBoost. For unsupervised learning, we are using clustering algorithms such as Kmeans, GMM and DBSCAN.

6 Conclusion

We can see that HOG is a good descriptor for object detection, and good performance can be achieved with Random Forest and linear SVM. One can expect even better performance with kernel SVM, if the computational complexity is not considered. DBSCAN relies on the density of data points so sparse datasets may not work well and lose efficiency on higher dimensional data sets such as this one. Convolutional Autoencoder should be a good model to compress image data. But it is difficult to train. From the result, we can see that it performs better with K-Means, SVM and Random Forest Algorithms. Table[2] shows that the performance of reduction/feature extraction method should be

evaluated by fixing a classification or clustering algorithm. For different classification or clustering algorithm, the best feature extraction method may vary. In general, supervised classification algorithm performs better than unsupervised clustering in our project. Based on the experiment results, HOG performs better than other two feature extraction methods. Neural Network performs best, it can classify image data without any dimension reduction. The test accuracy reaches 94.33%.

References

[1] Garcia, B. & Viesca, S.A. (2016) Real-time American Sign Language Recognition with Convolutional Neural Networks, *Convolutional Neural Networks for Visual Recognition, 2016*.

[2] S. Ameen, S. Vadera, A convolutional neural network to classify american 530 sign language fingerspelling from depth and colour images, Expert Systems.

[3] Zheng L, Lian B, Jiang A. (2017) Recent Advances of Deep Learning for Sign Language Recognition. 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)

[4] https://www.kaggle.com/datamunge/sign-language-mnist

[5] Nathan Halko,, et al. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. arXiv:0909.4061v2 [math.NA]

[6] Dalal, N. and Triggs, B., "Histograms of Oriented Gradients for Human Detection," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, San Diego, CA, USA Jonathan Masci , Ueli Meier , Dan Cireşan , Jürgen Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, Proceedings of the 21th international conference on Artificial neural networks, p.52-59, June 14-17, 2011, Espoo, Finland

[7] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. doi:10.1007/BF00994018.

[8] https://www.kaggle.com/ranjeetjain3/deep-learning-using-sign-langugage

[9] Mitchell, Ross; Young, Travas; Bachleda, Bellamie; Karchmer, Michael (2006). "How Many People Use ASL in the United States?: Why Estimates Need Updating" (PDF). Sign Language Studies (Gallaudet University Press.) 6 (3). ISSN 0302-1475. Retrieved November 27, 2012.