Project 2 - Practical Machine Learning
Comparison between two unsupervised methods

Alexandru Rusu - group 507

Dataset

For this project I used Lego Brick dataset (dataset link) which consists of 40000 unique images of 50 different lego parts. Each object is rendered from a different point of view angles: a complete 360 degrees sphere rotation.

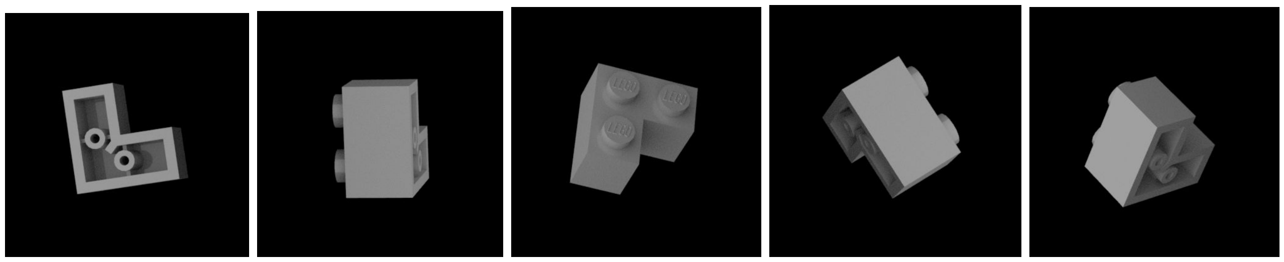

Fig. 1 Images for one type of Lego Brick datasets

Every image is stored as a **.png* file of size 400x400 pixels with the following name convention: id [brick name]_[order-number][L/R-left side of right side].png, by that the dataset is already labeled through file name convention.

An important aspect about this dataset is the fact that for every image, the background has been subtracted and replaced with a black one.

Solution proposed

In order to compare two unsupervised algorithms, I decided to do an image-clustering on the dataset and determine how unsupervised techniques could cluster this dataset.

For this image clustering task I used K-means with 50 clusters and Agglomerative Clustering algorithms with also 50 clusters and ward linkage.

For these algorithms to be able to cluster the images, I had to generate a feature set for every image. I took two approaches to do that. One of them was to simply generate a raw-pixels feature vector. For every image, I did a resize from 400x400 to 32x32 and flatten the image, in the end resulting a nparray of shape (3072,) -- 32x32x3 (RGB channel). The other one was to use a pre-trained model to extract features from images. For this, I used convolutional based VGG16 model from Keras. Comparing to raw-pixels vector, this time, images were resized to 224x224, the size of images from ImageNet.

After iterating through all the images, three lists are generated containing labels, image features and image path. Because it is an unsupervised task, entire dataset was used to test the models.

First method, by using raw-pixels vector was a bad method of data fetching for the clustering algorithms because the lego bricks are colored in gray-scale and clustering algorithms would see only random pixels and nothing useful.

The second way of data preprocessing was a better approach and this time, the algorithms really got important features to cluster. The down-side about this method is the time elapsed for preprocessing. Because images are big, 224x224, and there are a lot of samples in the dataset, to save time for testing, the numpy array resulted from feature extraction steps were saved in local txt files. The model used was a pre-trained one and there was nothing to improve about this step.

After all features have been extracted and saved locally, I started to compare different algorithms of clustering and kept the two algorithms mentioned earlier.

The Output

After I run both algorithms, I moved every clustered image in a separate folder based on the predicted cluster. The images were not clustered the same as they are by the input label, but how they were actually clustered is quite interesting. The algorithms chose to split them by their particularities: one cluster contains images of round shape lego bricks, other cluster contain back-view bricks with two holes, and so on. For the bricks that are very different from the others, the algorithms clustered them very well, even if the rotation angle was from the side, or upward. There are also some clusters that may seem to contain random bricks, but after a closer look, it became obvious that they share the same big flat area on one side.

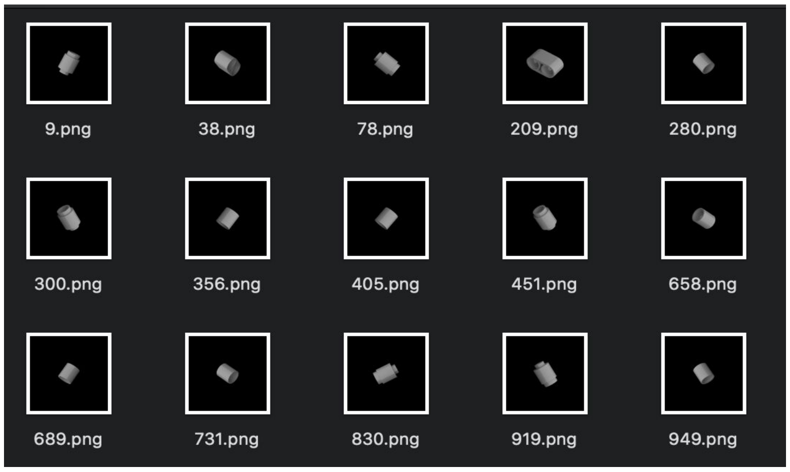

Fig. 2 One cluster determined by K-means

Conclusion

In this test, both unsupervised algorithms did a great job, but the factor that boosted them were the pre-processing step where I used VGG16 network. In conclusion, even though cluster algorithms may seem pretty easy and weak, if they get the right input they can do an amazing job and surprise you by the way they chose to group things.