Unsupervised Image Classification

By Royce Schultz

Abstract

Unsupervised learning ultimately boils down to some form of clustering. Popular methods include:

- K-means
- Agglomerative Clustering
- Affinity Propagation

Performance relies on good feature choice. PCA is an example of automatic feature detection. Using eigenvalues, PCA determines the most variant directions, and transforms the data according to these vectors, however this process is far too slow for large data structures like images with length over 3000 for very small images (32x32x3).

Machine learning can be useful to determine effective feature vectors. Once trained, the model can quickly encode new images. This project covers some machine learning models that can generate embedded vectors.

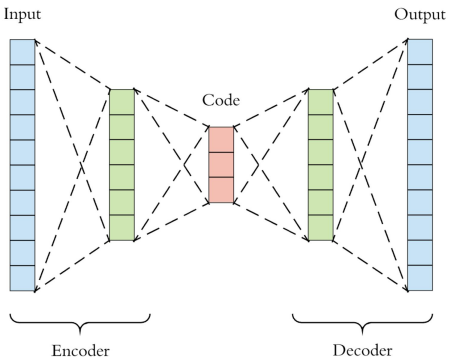
Attempt 0: Control Test

A baseline test is performed by downscaling images to a reasonable vector size. Clustering on these vectors still performs better than random guessing because classes often share common colors like airplanes or boats on a blue background or a green frog.

Attempt 1: Auto-Encoder

To train an auto-encoder, the model takes an image as input and tries to recreate the images after passing the data through a bottleneck layer. This is a form of compression. In practice, Images retain much of the visual information with an encoding just 10% the size of the original image. However, jpeg compression is more efficient for data storage purposes.

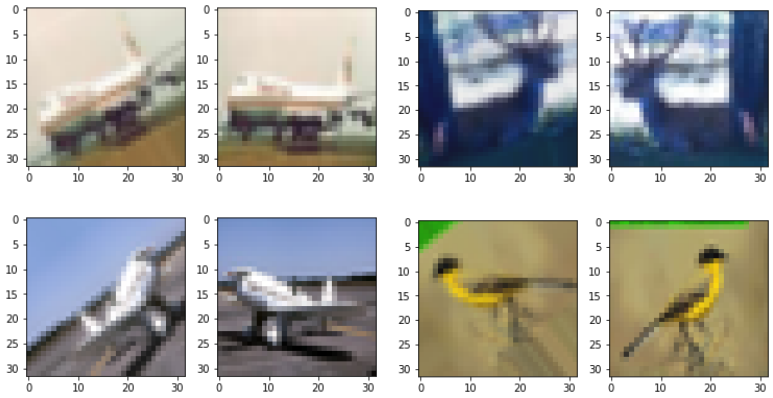
The model ‘memorizes’ common shapes so it can recreate the image. Images of the same underlying class often share many visual shapes which should improve clustering performance.

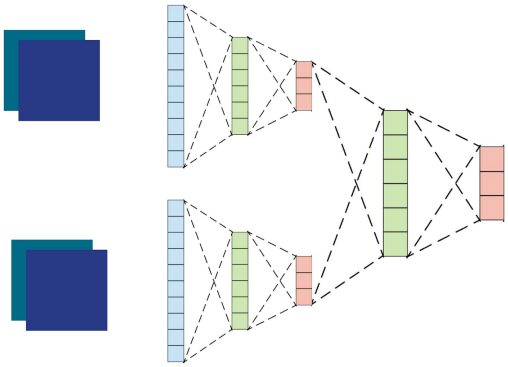


The decoding step may be detrimental to results because the encoding must include location information. An object in the bottom corner is encoded differently from the same object in the top corner.

Attempt 2: Representational Learning

This method attempts to encode different representations of the same object as a more similar vector, addressing the locational information problem of auto-encoders. Images are generated in a semi-supervised manor by rotating, flipping, or scaling images. The model trains by classifying generated pairs as '1' and randomly paired 'mis-matches' as '0'.

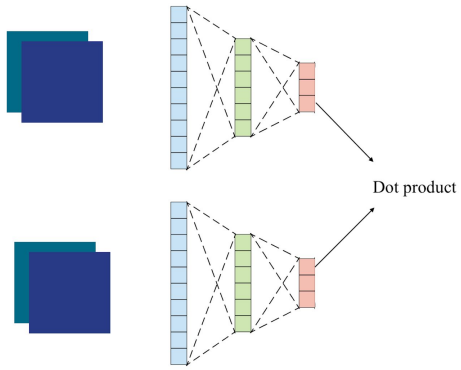




This is the same structure used in ‘identity recognition’ where the model is provided 2 images of faces and must determine if they are the same person. In that case, the extra ‘comparison’ layers may be necessary, however it may not be the best solution for unsupervised learning as it is not a direct similarity measure. The encodings produced are not guaranteed to be close to one another, only that it is ordered according to however the comparison has learned to recognize them.

Attempt 3: Similarity Measures

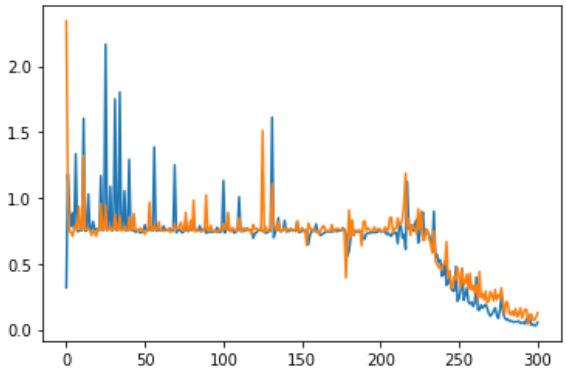
This model uses a more reliable similarity measure, a dot product. Regularization is necessary to produce stable vectors. If a normalized dot product is used (cosine distance), the encodings will shrink to near 0 as the length of the vector is not accounted for in the loss.



The regularization method I used produced very sparse vectors and somewhat unstable results. A more clever loss function should produce more reliable results.

The vanishing gradient problem.

In practice, many more intermediate convolutional layers are used than what is pictured. When many layers are randomly initialized, the image data after just a few layers becomes random noise. Additionally, the gradient shrinks as it backpropages from the output layer. Eventually the backpropagation will create order in the noise, but it may take a significant number of training cycles.



This figure depicts the loss during training (on very small batches). The initial flat section is where the model is guessing 0.5 similarity regardless of input. After many training cycles, the early layers eventually begin to form some sort of order and the model begins to make more specific guesses.

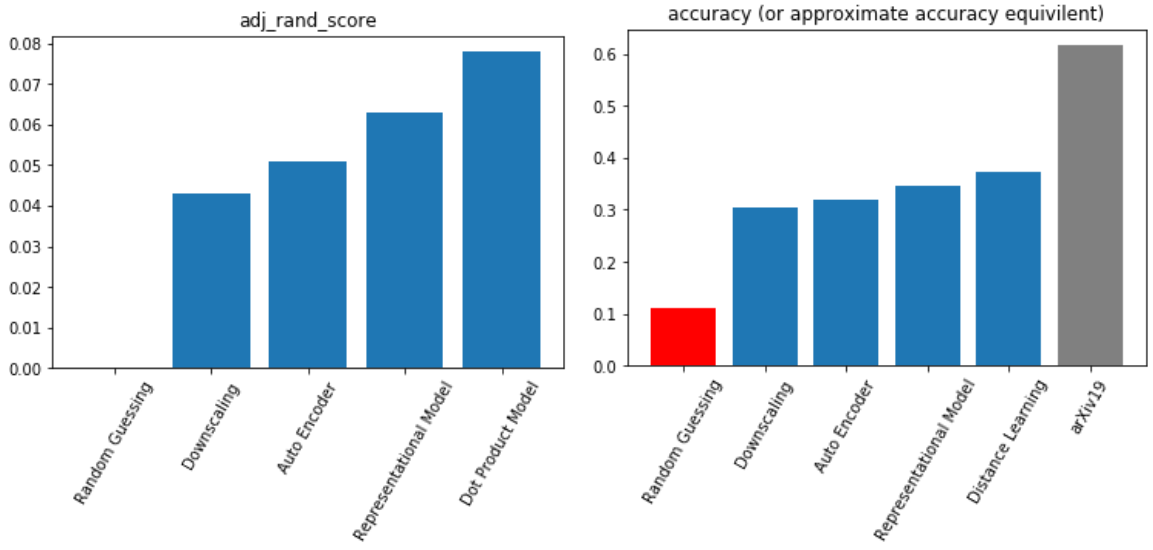
Results

Dataset

I used the CIFAR-10 dataset. It contains 60,000 images of vehicles and animals in 10 labeled categories. The model did not see these labels during training.

Scoring

Encodings are clustered and then scored using sklearn's adjusted rand score function. Random simulation is used to approximate an equivalent accuracy for comparison with arXiv19. A sample dataset is generated to a prescribed accuracy and then scored using the adjusted rand score.



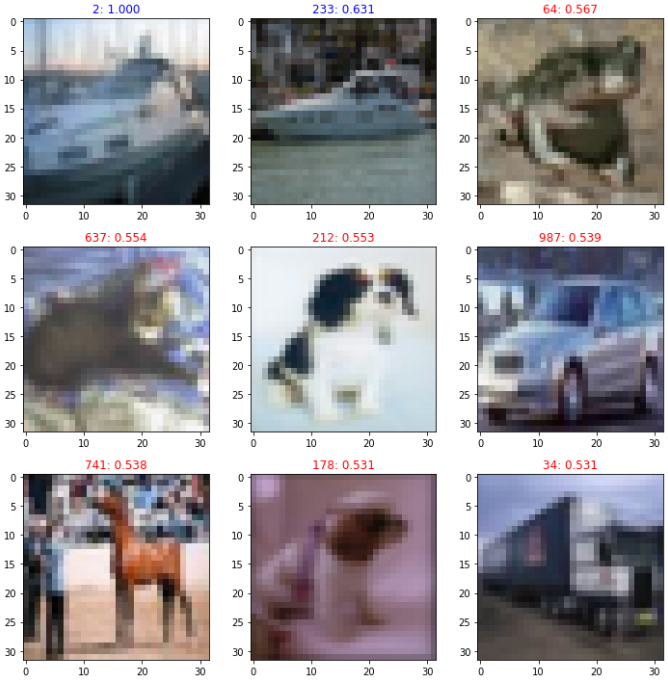
While I got the best score using the dot product model, training was most stable using the representation model.

In the following figures, the images shown were the most similar to the target image which is often (but not always) in the top left corner. The images were compared on a subset of the data because calculating all pairwise similarities is very slow.

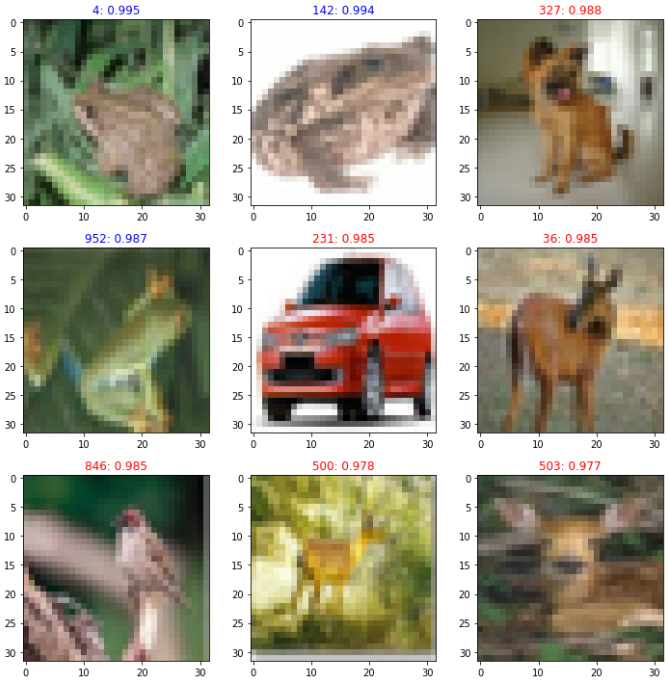
The model identified many examples of boats, many in similar orientations.

The target image here is actually number 9 (top, middle). The car in the first position was identified as more similar than the image itself. Not ideal, although it did identify many cars.

This set failed to identify many planes, but instead matched against things with blue backgrounds. However, birds are still pretty close to the target image.



This example failed to identify other boats, unlike the first example.



This example identified a few frogs, but also matched to other animals with similar colors and for some reason a bright red car.

The models could often match trucks, cars, or airplanes, but matching the animal classes performed significantly worse. Identifying certain images of animals at this resolution is a difficult task for humans. Perhaps greater resolution photos could help as the nuanced features like pointy ears vs floppy ears may be easier to identify, but then the model is more difficult to train because it has more weights.

Conclusions

Machine learning does improve clustering results over random guessing and the downscaling control test, however results are still not particularly good.

Binary similarity may not effectively describe the data. A plane and bird share many features so a ‘0’ similarity is not appropriate. ArXiv19 uses a matrix multiplication to achieve a more 'feature-full’ similarity metric.