Assignment 2 - K-means Clustering and Auto-Encoder

Sagar Sonawane - ssonawan@buffalo.edu

14th November 2021

1 Assignment Overview

This assignment aims to perform unsupervised learning on CIFAR - 10 dataset. In part 1 of the assignment implementation of K-means Clustering from scratch is to be performed and evaluated by using ASC and Dunns Index. In part 2 of the assignment, the implementation of K-means clustering is to be performed on the encoded representation generated by an Auto-Encoder model. Evaluation of part 2 is to be done by using ASC and Dunns Index.

2 Dataset

The Dataset used for this assignment is the CIFAR - 10 dataset, consisting of 60,000 images of 10 classes. There are 50,000 images in the training dataset and 10,000 in the test dataset.

The dataset contains the following classes.

1. Airplane
2. Automobile
3. Bird
4. Cat
5. Deer
6. Dog
7. Frog
8. Horse
9. Ship
10. Truck

3 Python Editor

I have made use of Jupyter Notebook IDE for the implementation of the program.

PART 1:

4 Implementation

I have downloaded the CIFAR - 10 dataset and the test set has been implemented in the K-means model.

4.1 K-means Clustering

K-means is an unsupervised learning technique that aims to divide the “n” number of observations into some “k” number of clusters based on how far the observation is from the mean of a cluster. K-means algorithm can be implemented in four steps:

- Define initial k clusters, and their centers $c = c_1, c_2, c_3 ... c_k$.
- For every observation x_i, calculate the distance between xi and the mean(center) of the cluster, and classify x_i into the cluster with the shortest distance.
- Then for every cluster a_k ,recalculate the center, $a_k = 1/c_i \sum x$
- Repeat the two steps above to reach a terminating condition.

4.2 Accuracy

The Quality of the clusters is checked with the help of the Silhouette score and Dunns Index score. These two metrics are used to check how accurately did the model create clusters based on the image features(observations).

4.2.1 Silhouette Score

The Silhouette score coefficient is calculated using the mean intra-cluster distance “a” and the mean nearest-cluster distance “b” for each point.

The Silhouette Coefficient for a sample is (b - a) / max(a, b).

Note that Silhouette Coefficient is only defined if number of labels is

2 <= n_labels <= n_samples - 1.

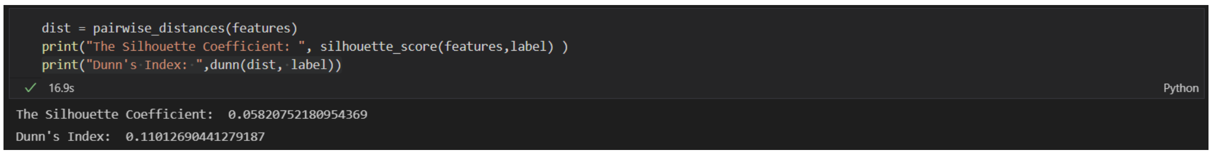
The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

4.2.2 Dunns Index Score

The Dunn Index (DI) is a metric for evaluating clustering algorithms. It is an internal evaluation scheme, where the result is based on the clustered data itself. The Dunn Index aims to identify the set of clusters that are compact with low variance between the members. The higher the DI score the better is the cluster.

5. Conclusion

After 200 Iterations with 10 clusters, the model gave an output silhouette score of 0.0582 and Dunns Index score of 0.1101.



```
dist = pairwise_distances(features)
print("The Silhouette Coefficient: ", silhouette_score(features,label) )
print("Dunn's Index: ",dunn(dist, label))
```

✓ 16.9s
Python

```
The Silhouette Coefficient:  0.05820752180954369
Dunn's Index:  0.11012690441279187
```

Fig 1. The output of Part 1

PART 2:

4 Implementation
The training data and the test data from the CIFAR - 10 dataset have been made use of for the training and the validation of the autoencoder respectively.

4.1 Auto-Encoder

An Auto-Encoder is a type of artificial neural network which is used to learn efficient coding of unsupervised data. Auto-Encoder learns by first encoding the unsupervised input data following with validation and refinement of the model by trying to reconstruct(decode) the encoded input.
Auto-Encoder has two parts - Encoder and Decoder.

4.1.1 Encoder

The encoder is the part where the autoencoder learns a representation for some input data, usually by reducing the dimensions as a way to train to ignore the noise.

There are two convolutional layers used in the encoder. The first layer has 16 neurons with the kernel size of (3,3) and strides of size 2 and the second layer has 8 neurons with the kernel size of (3,3) and strides = 2. Both the layers use the RelU activation function.

(Strides will move the kernel over a given number of pixels thus, increasing or decreasing the dimensions of the input image)
The encoded output is then passed as an input for the K-means model.

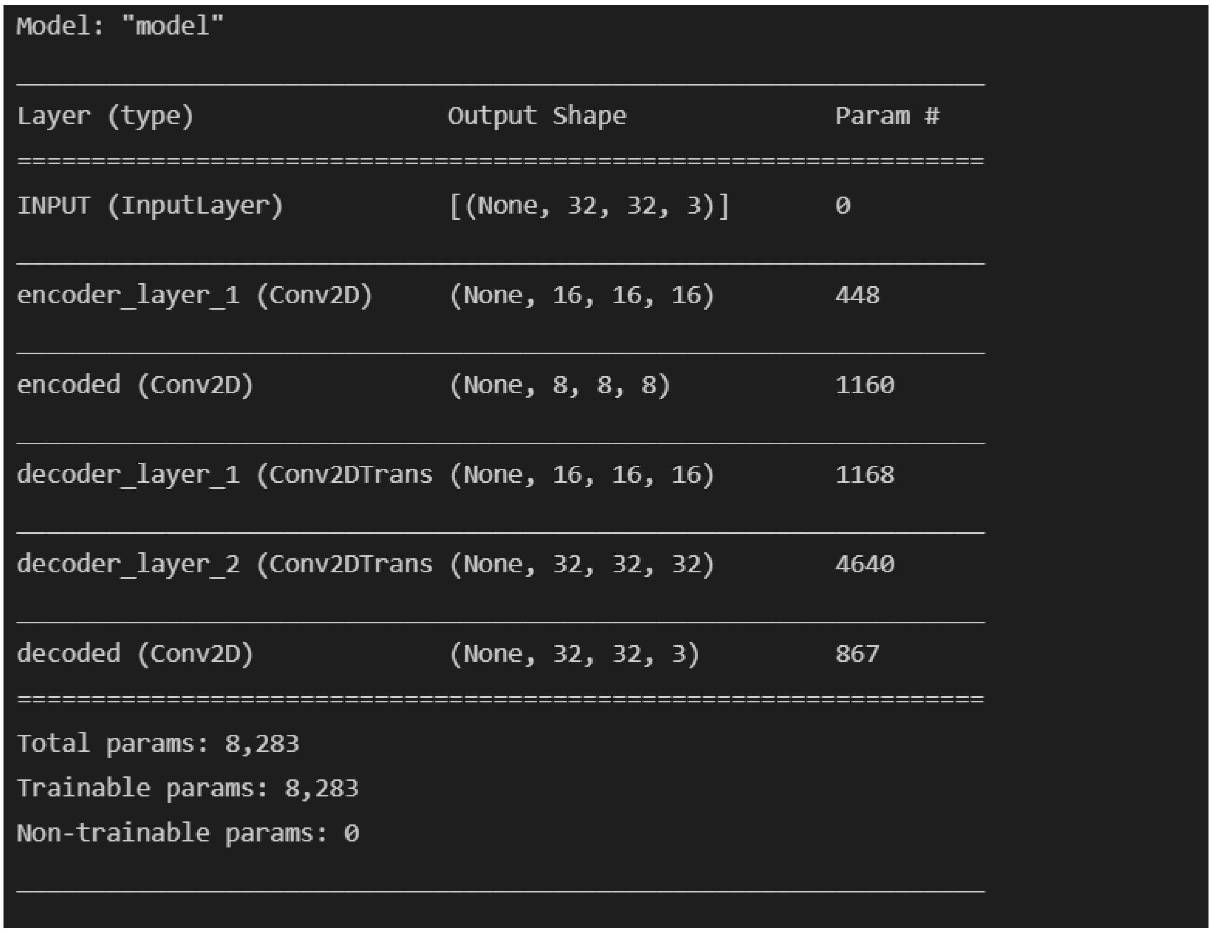
4.1.2 Decoder

The decoder is the part where the autoencoder regenerates the input data from the encoded output to evaluate the efficiency of the encoding.

I have made use of 3 layers in the decoder. The first two layers have 16 and 32 neurons resp. with a kernel size of (3,3) and strides = 2 with activation function as RelU. These two layers are transposed convolutional(deconvolutional) layers which change the shape of the output back to its original input. The third layer is a convolutional layer, with 3 neurons and has the sigmoid activation function.

The decoded images are then plotted to be compared with the original input.

4.1.3 Implementation of Auto-Encoder
Training of Auto-Encoder was done for over 5 epochs with 128 batch size with x_train as training data and x_test as validation data



```
Model: "model"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
INPUT (InputLayer)          [(None, 32, 32, 3)]       0

_________________________________________________________________
encoder_layer_1 (Conv2D)    (None, 16, 16, 16)        448

_________________________________________________________________
encoded (Conv2D)            (None, 8, 8, 8)           1160

_________________________________________________________________
decoder_layer_1 (Conv2DTrans (None, 16, 16, 16)       1168

_________________________________________________________________
decoder_layer_2 (Conv2DTrans (None, 32, 32, 32)       4640

_________________________________________________________________
decoded (Conv2D)            (None, 32, 32, 3)         867
=================================================================
Total params: 8,283
Trainable params: 8,283
Non-trainable params: 0
_________________________________________________________________
```

Fig 2. Auto-Encoder structure

4.2 K-means implementation

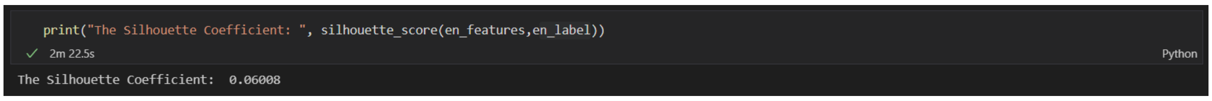
The encoded output from the autoencoder is first reshaped and then passed into the previously created K-means model as the input for 100 iterations with 10 clusters.

4.3 Evaluation

The efficiency of this K-means model with the encoded input is evaluated by using the Silhouette Coefficient Score.

5. Conclusion

After 100 Iterations with 10 clusters, the model gave an output silhouette score of 0.06008.



```
print("The Silhouette Coefficient: ", silhouette_score(en_features,en_label))
✓ 2m 22.5s                                                              Python
The Silhouette Coefficient:  0.06008
```

Fig 3. The output of Part 2

The decoded output from the Auto-Encoder has been plotted along with the original input images.

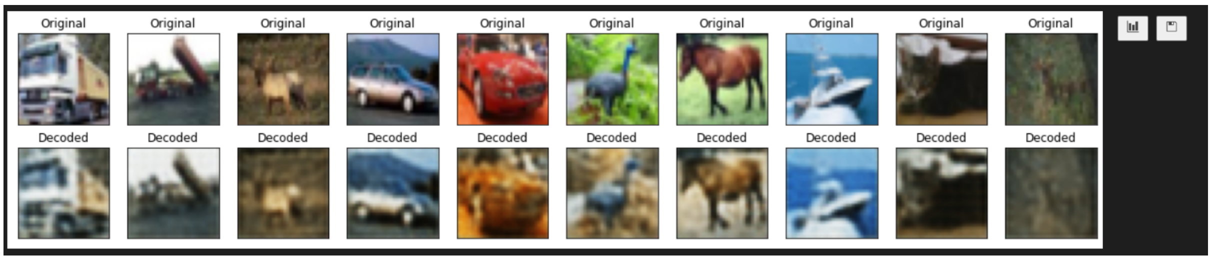


Fig 4. Plotting of original input vs decoded output