CSE 474/574: Introduction to Machine Learning Project 2

Vidushi Sharma
Department of Computer Science
University at Buffalo
Buffalo,NY 14260
vidushis@buffalo.edu

Abstract

Neural Networks are the programmable patterns that help to solve complex problems and bring the best achievable output.The task of the project is to perform unsupervised learning on Cifar 10 dataset. There are two tasks, the first is to perform K-means clustering on the raw data from scratch. The second is to perform Kmeans clustering on a representation generated by the Auto-Encoder method using library functions.

1. Introduction

Supervised vs Unsupervised Learning:

In **Supervised Learning**, the algorithm "learns" from the training dataset by iteratively making predictions on the data and adjusting for the correct answer. While supervised learning models tend to be more accurate than unsupervised learning models, they require upfront human intervention to label the data appropriately.

Unsupervised Learning models, in contrast, work on their own to discover the inherent structure of unlabeled data. Note that they still require some human intervention for validating output variables.

2. DataSet

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

In our task, we use the testing dataset as our Training Dataset. Hence our new training dataset for the problem has 10,000 images of 32*32 size.

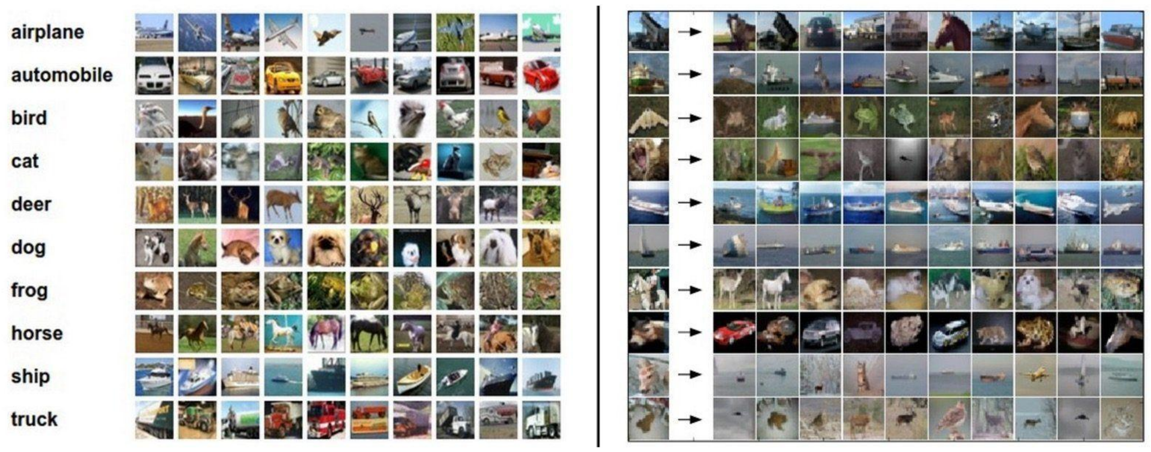


Figure1: Classification of Images present in the Cifar Dataset in 10 categories[1]

3. K Means

3.1 Algorithm

1. Initialise K points randomly from the dataset and set these as the initial centroids.
2. All the data points should be assigned in K groups based on their distance from K clusters, It involves assigning the data point to the nearest cluster.
3. Take the mean of all the points present in the cluster and make that mean as the centroid of the respective cluster.
4. Perform step 2 and 3 until all clusters are found.

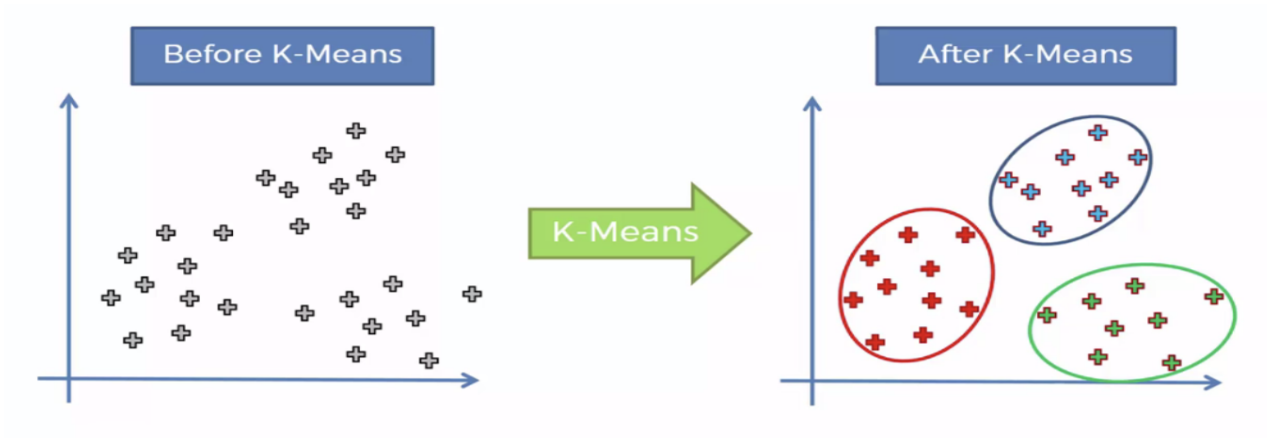


Figure 2: Clustering of random data points into defined clusters=3 by using KMeans [3]

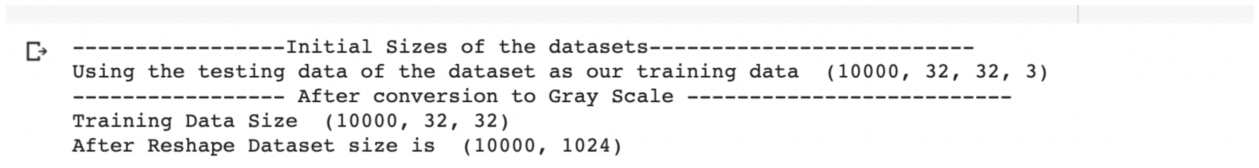
3.2 Data Preprocessing:

Data Normalisation involves adjusting values measured on different scales to a common scale. When dealing with dataframes, data normalization permits to adjust values referred to different columns to a common scale. This operation is strongly recommended when the columns of a dataframe are considered as input features of a machine learning algorithm, because it permits to give all the features the same weight.

We make sure that we normalise the data initially to remove any outliers by using the following:

```
#Preprocessing the Dataset : Normalising
train_X = train_X.astype('float32') / 255
```

In our example the initial shape of the training data is 10,000 * 32 * 32 * 3 . We use the technique of flattening to flatten the data to the dimension 10,000 * 32 * 32 . And then reshape the data to fit in the dimension of 10000 * 1024 .



```
-----------------Initial Sizes of the datasets--------------------------
Using the testing data of the dataset as our training data  (10000, 32, 32, 3)
----------------- After conversion to Gray Scale --------------------------
Training Data Size  (10000, 32, 32)
After Reshape Dataset size is  (10000, 1024)
```

4. AutoEncoder

An Autoencoder is a deep neural network which tries to learn the function *f(x)* ≈ *x* or in other words, it learns to copy it’s input to its output. Autoencoder learns by minimizing the loss function. In the code we have used, the loss function is a mean squared error and the optimiser as “Adam Optimiser”. The output is a set of reconstructed grayscale images. The loss function penalizes the autoencoder for not being able to reconstruct the grayscale version of the image.

An AutoEncoder comprises of two components:

1. **Encoder**:
   The Encoder is responsible for compressing the input image into latent space (lower dimensionality).
2. **Decoder**:

A Decoder is responsible for re-constructing the image back from the latent space (lower dimensionality) to the almost original image.

4.1 Data Preprocessing:

Since we have image data which has 3 channels of depth i.e. RGB, it becomes sometimes difficult to run a Neural Network on such data. In such cases in order to apply Convolution properly we convert the input data into Gray Scale Images.

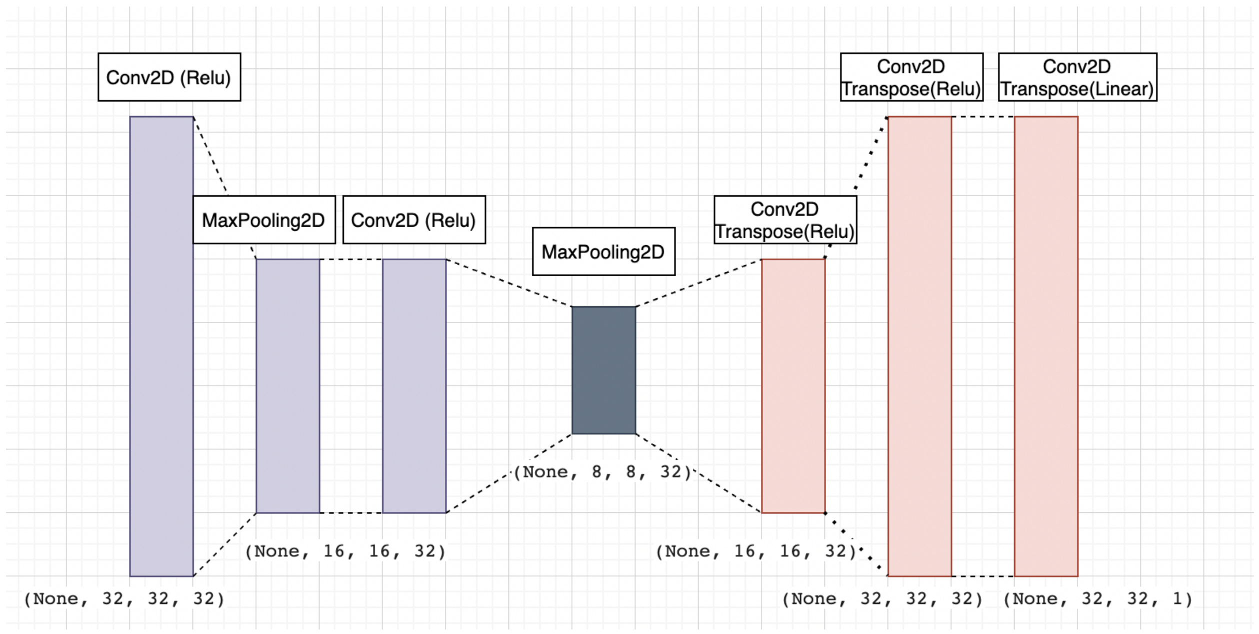
4.2 Architecture:

Conv2D Layer:

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. [6]

MaxPooling2D Layer:

Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. [6]



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 1)]       0

 conv2d (Conv2D)             (None, 32, 32, 32)        320

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 conv2d_transpose (Conv2DTra  (None, 16, 16, 32)       9248
 nspose)

 conv2d_transpose_1 (Conv2DT  (None, 32, 32, 32)       9248
 ranspose)

 conv2d_2 (Conv2D)           (None, 32, 32, 1)         289
```

5. Findings

We will be using the below metrics to analyse the performance of our tasks:

Silhouette score:
Silhouette score is used to evaluate the quality of clusters created using clustering algorithms such as K-Means in terms of how well samples are clustered with other samples that are similar to each other. The Silhouette score is calculated for each sample of different clusters.

Dunn Index:
Dunn index to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated, where the means of different clusters are sufficiently far apart, as compared to the within cluster variance.

5.1: Implementing K Means

No_of_Clusters	No_of_Iterations	Dunn_Index	Silhouette Score
10	100	0.0912171	0.060050305

```
Requirement already satisfied: validclust in /usr/local/lib/python3.7/dist-packages (0.1.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from validclust) (1.1.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from validclust) (3.2.2)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.7/dist-packages (from validclust) (1.19.5)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from validclust) (0.22.2.post1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (from validclust) (0.11.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from validclust) (21.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->validclust) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->validclust) (1.3.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->validclust) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->validclust) (0.10.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib->validclust) (1.15.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas->validclust) (2018.9)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->validclust) (1.4.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->validclust) (1.0.1)
Dunn Index  is  0.0912171
```

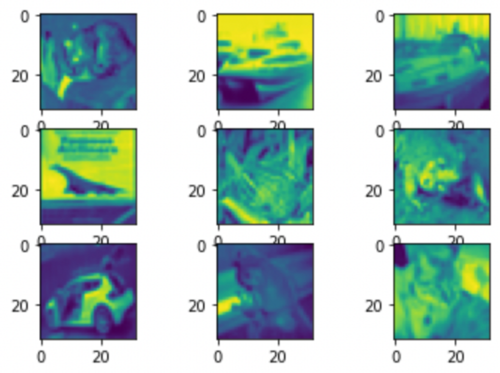
5.2: Implementing Autoencoder

Epochs	Batch_Size	Silhouette Score
10	100	0.043691194095152396

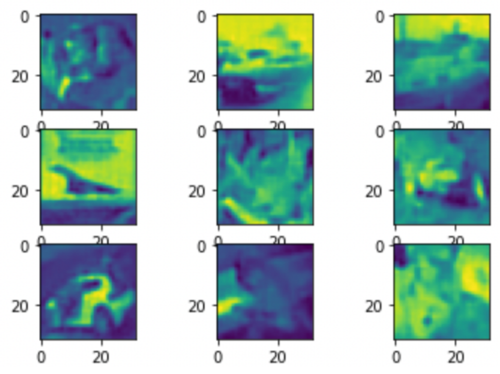
CALCULATING SILHOUETTE_SCORE

```
[ ] silhoutte_score_part2 = silhouette_score(x_test.reshape(10000, 32*32), kMeans_Output)
    print("Clusters = 10 , The silhouette_score is :", silhoutte_score_part2)

Clusters = 10 , The silhouette_score is : 0.043691194095152396
```



```
Clusters = 10 , The silhouette_score is : 0.043691194095152396
--------------------Comparing Test Images with Reconstructed Images--------------------------
```



6. References

1. https://blog.kickview.com/training-a-cnn-with-the-cifar-10-dataset-using-digits-5-1/
2. https://www.cs.toronto.edu/~kriz/cifar.html
3.https://towardsdatascience.com/k-means-clustering-identifying-f-r-i-e-n-d-s-in-the-world-of-strangers-695537505d
4. https://www.geeksforgeeks.org/dunn-index-and-db-index-cluster-validity-indices-set-1/
5. https://blog.keras.io/building-autoencoders-in-keras.html
6. https://keras.io/api/layers/pooling_layers/max_pooling2d/