PCA-Based animal recognition

Name:- Aparna Krishna Bhat

ID:- 1001255079

Introduction

Principal Component Analysis (PCA) is a linear dimensionality reduction technique that can be utilized for extracting information from a high-dimensional space by projecting it into a lower-dimensional sub-space. It tries to preserve the essential parts that have more variation of the data and remove the non-essential parts with fewer variation.

Dimensions are nothing but features that represent the data. For example, A 32 X 32 image has 1024 picture elements (pixels) that are the dimensions or features which together represent that image.

One important thing about PCA is that it is an Unsupervised dimensionality reduction technique, that will cluster the similar data points based on the feature correlation between them without any supervision (or labels).

In this project I have integrated PCA with KNN that not only reduce the data dimensionality to speed up the calculation of KNN, but also reduce redundant information while retaining the important information which affects the performance of KNN prediction. Further, to investigate and compare the performance of PCA I have integrated it with a deep learning model. Finally, deep learning model without PCA have been implemented to see the performance difference between using PCA and without using it.

Datasets used

- ❖ The **CIFAR-10** (Canadian Institute for Advanced Research) dataset consists of 60000 images each of 32x32x3 color images having ten classes, with 6000 images per category. The dataset consists of 50000 training images and 10000 test images. The classes in the dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.
- ❖ **Animals-10** which contains about 28K medium quality animal images belonging to 10 categories: dog, cat, horse, spider, butterfly, chicken, sheep, cow, squirrel, elephant.

Where all you can apply PCA?

1) Data Visualization
2) Speeding Machine Learning algorithms.

Steps followed for PCA implementation

Dataset Used:- CIFAR-10

- Checked the maximum and minimum values of the CIFAR-10 training images and normalize the pixels between 0 and 1 inclusive.
- Created a DataFrame that holds the pixel values of the images along with their respective labels in a row-column format.
- Reshaped the image dimensions from three to one (flatten the images).
- Created the PCA method and pass the number of components as two and apply **fit_transform** on the training data.
- Converted the principal components for each of the 50,000 images from a NumPy array to a panda DataFrame.
- Found out the amount of information or variance the principal components hold.

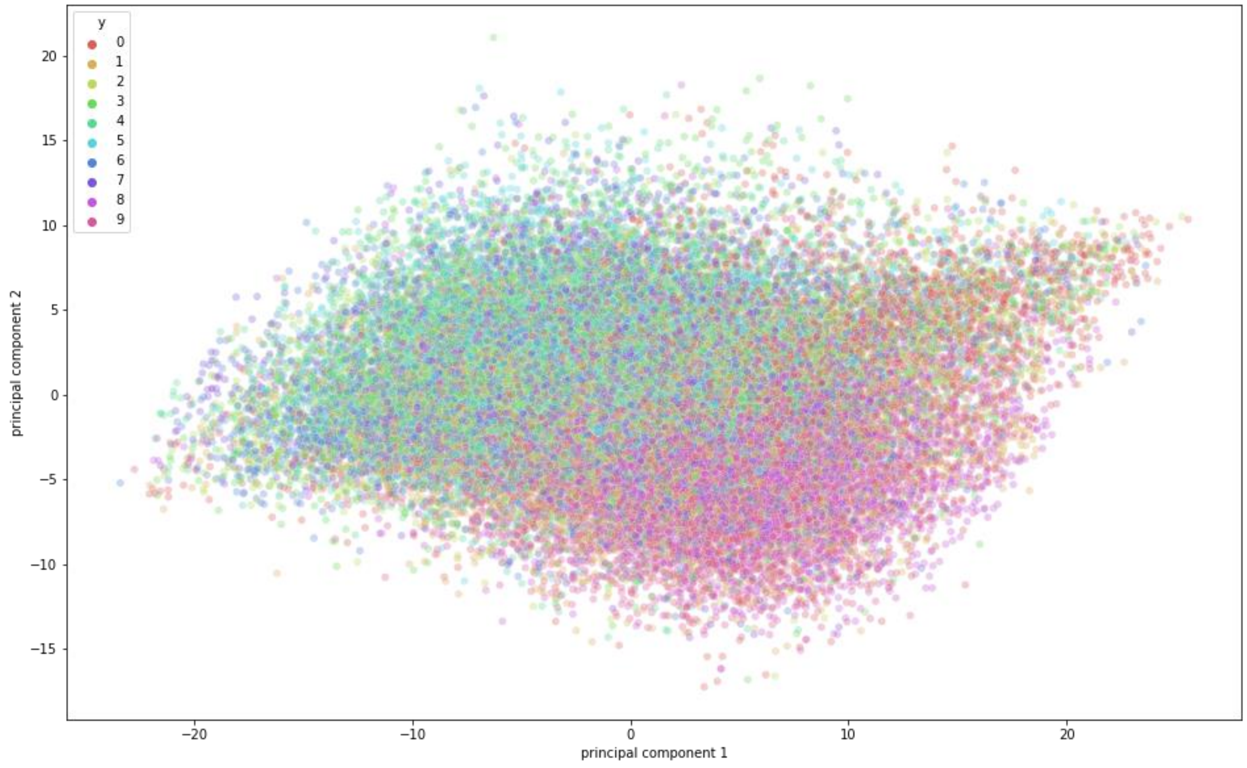
PCA relies on principal components to represent what's going on with your data. In layman's terms, the three dimensions are nothing more than the three Principal Components that capture (or hold) the majority of the variance (information) of your data when projected into a lower dimension (assuming three dimensions) from a higher space. The magnitude and direction of principal components are equally important. The magnitude denotes the amount of variance the Principal Component captures when projected onto that axis, and the direction denotes which principal axes the data is generally spread out or has the most variance across. A straight line represents the principal components, and the first principal component contains the largest variance in the data. The next principal component is orthogonal to the previous one and has a lower variance.

Each principal component represents a percentage of total variation captured from the data.

Principal components used for this project:- 11 and variation per principal component obtained in given below

Explained variation per principal component: [0.2907663 0.11253144 0.06694414 0.03676459 0.03608843 0.0280923 0.02712992 0.02167162 0.02064641 0.01438001 0.01310559]

PCA visualization obtained (plotting 2 principal components)



PCA with KNN (K- nearest neighbors)

KNN can be summarized as below:

1. Computes the distance between the new data point with every training example.
2. For computing the distance measures such as Euclidean distance, hamming distance, minkowski or Manhattan distance will be used.
3. Model picks K entries in the dataset which are closest to the new data point.
4. Then it does the majority vote i.e the most common class/label among those K entries will be the class of the new data point.

Results obtained are as follows:

Have used KNeighborsClassifier from sklearn

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',metric_params=None, n_jobs=None, n_neighbors=12, p=2,weights='uniform')

Current accuracy is 38.34% for K=12. Applied for CIFAR-10 dataset consisting of 50000 training images and 10000 test images.

Confusion matrix and Classification report obtained are as follows

```
[[564   8 101  15  57   3  35  10 199   8]
 [ 94 266  90  28 133  17 115  18 203  36]
 [104   3 427  41 230  27 108  12  43   5]
 [ 55  13 199 170 200  82 201  25  41  14]
 [ 70   2 230  24 525   6  82  22  38   1]
 [ 47   8 191 110 195 232 146  16  48   7]
 [ 14   0 199  31 259  16 453   3  22   3]
 [ 74   5 156  50 250  48  76 272  55  14]
 [111  13  42  33  55  12  25  11 680  18]
 [132  70  62  31 104  21  89  24 222 245]]
              precision    recall  f1-score   support

           0       0.45      0.56      0.50      1000
           1       0.69      0.27      0.38      1000
           2       0.25      0.43      0.32      1000
           3       0.32      0.17      0.22      1000
           4       0.26      0.53      0.35      1000
           5       0.50      0.23      0.32      1000
           6       0.34      0.45      0.39      1000
           7       0.66      0.27      0.38      1000
           8       0.44      0.68      0.53      1000
           9       0.70      0.24      0.36      1000

    accuracy                           0.38     10000
   macro avg       0.46      0.38      0.38     10000
weighted avg       0.46      0.38      0.38     10000
```

PCA with deep learning model

- Created an instance of the PCA model.
- You may also specify how much variance you want PCA to capture. Have given PCA model a value of 0.9, which means it will keep 90% of the variance and use the number of components available to capture 90% of the variance.
- ***To achieve 90% variance, the dimension was reduced to 99 principal components from the actual 3072 dimensions.***
- Deep learning hyperparameters used in the sequential model are set as batch_size = 128, num_classes = 10, epochs = 20, Activation functions of relu and softmax.
- The time taken for training each epoch was just 1 second on a GPU. The model did a decent job on the training data, ***achieving 95% accuracy while it achieved only 56% accuracy on the test data***. This means that it overfitted the training data.

Model

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_43 (Dense)             (None, 1024)              102400    
_________________________________________________________________
dense_44 (Dense)             (None, 1024)              1049600   
_________________________________________________________________
dense_45 (Dense)             (None, 512)               524800    
_________________________________________________________________
dense_46 (Dense)             (None, 256)               131328    
_________________________________________________________________
dense_47 (Dense)             (None, 10)                2570      
=================================================================
Total params: 1,810,698
Trainable params: 1,810,698
Non-trainable params: 0
```

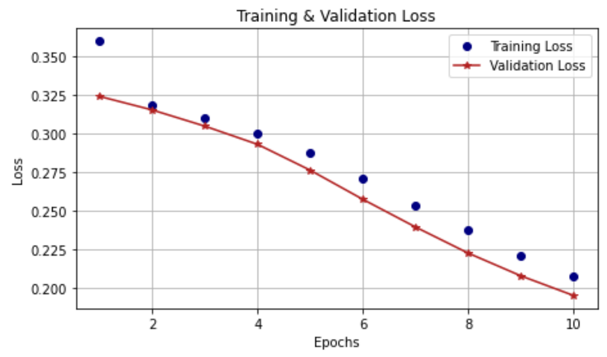
Animal-10 dataset CNN classification Without PCA

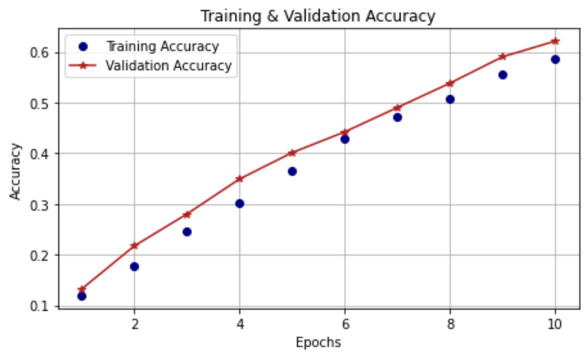
- Using VGG16-transferlearning model.
- Each epoch takes around 63 seconds of GPU time.
- Epochs=10, Train accuracy=0.58550, Validation accuracy=0.62100
- loss='binary_crossentropy',optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),metrics=['accuracy’]

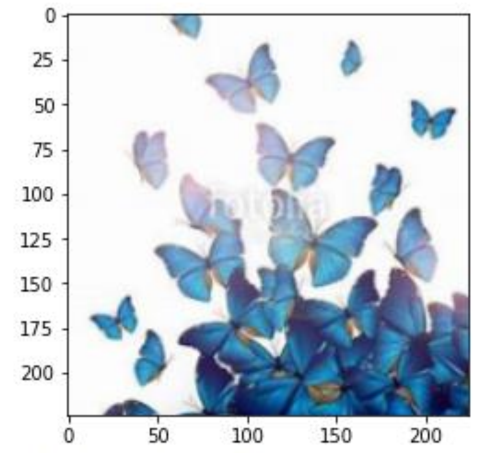
- Activation function used are relu and softmax.

Results obtained are as follows:

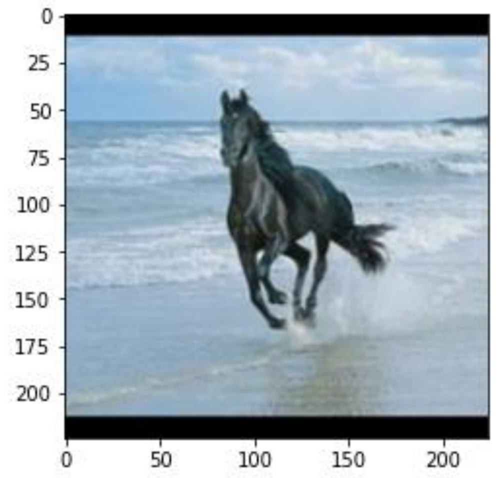
```
CNN: Epochs=10, Train accuracy=0.58550, Validation accuracy=0.62100
```



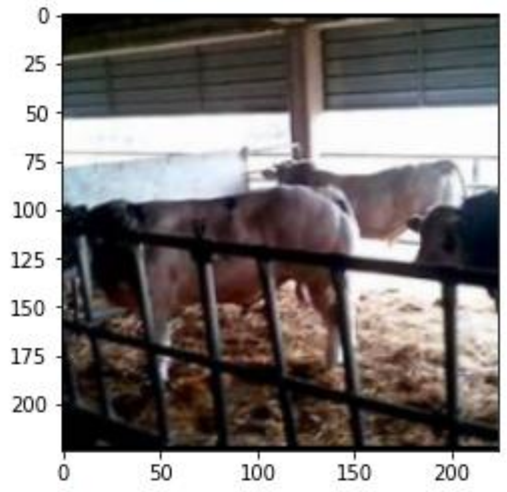




```
Actual  : butterfly
Predict : butterfly
```



```
Actual  : horse
Predict : horse
```



```
Actual  : cow
Predict : horse
```

Insights and key Takeaways

- PCA is a powerful algorithm which removes the correlated features and improves the performance of the algorithm.
- PCA Reduces overfitting.
- PCA with classification algorithm like KNN works well(depending upon the dataset and size of the dataset).
- PCA with deep learning model gives good accuracy and performance with faster training and execution time.
- Deep learning model without PCA also gives good accuracy but at a cost of very high learning/ training rate.

Experience and learning

When compared to clean and organized data, working with image classification is a different ball game. The data is processed and flattened as pixels. The flattened pixel collection is used to represent each pixel's details. The image quality and pixel density matrix have a significant impact on the classifier's accuracy. Since the animal data that was used was so vague. Backgrounds are used in the animal pictures. The aforementioned contributes to the classification disparity and degrades the efficiency of the classifier in use.