# Chapter - 1

# Introduction

## 1.1 General Introduction

### 1.1.1 Purpose

The goal of “Visual Category Recognition” is to predict the categories of the input image with the help of its features that can be extracted via visual cues like: shape and texture. There are various approaches for solving this problems such as k nearest neighbor(KNN), Adaptive boost (Adaboosted), Artificial Neural Network (NN), Support Vector Machine (SVM). We will be implementing a hybrid framework that possesses the advantages of both KNN and SVM techniques for classifying image in this project. The focus will be on “categories” rather than specific objects as we will be using a dataset containing images for 101 categories(Caltech 101 dataset).

The project will initially take the number of images as its training dataset and will provide the category of the input image as an output.

### 1.1.2 Project Scope

“Visual Category Recognition” comes under the field of Computer Vision. The category of an given object can not be predicted deterministically due to the similarities in the difference objects. For example, a snake on the river bed can be confused with the alphabet 'S'.

This project can be used for surveillance of the public places for the suspicious items. Moreover, an improved category recognizer when fitted into a robot can be used to manage the items in the industry. For example, counting each item present in the warehouse of the industry and removing the defective / unwanted materials from the wareouse. This will save both time, space and human resource.

Currently, this project is limited to identify the objects present in the 101 Caltech Data Set.

### 1.1.3 Operating Environment

This project is currently run on a computer device in either the terminal or by using the *jupyter-notebook* which acts as an environment to run the code snippets one by one. With the help of jupyter-notebook, one can easily compiler the currently written code and find out the bugs without affecting the previously written code.

In the terminal the project can be run by using the command

```
python vcr.py
```

from the director where the Caltech Data-Set is present. This ensures that the project gets the training data images and data sets for recognition. The advantage of using a jupyter-notebook than running it with python compiler everytime is that the value of the variables is preserved and the user have the freedom to run the code fragments according to their wish. You can run a code fragment f1 which is written after another fragment f2 which you don;t want to run without running the fragment f2.

### 1.1.4 Design & Implementation Constraints

- **Size Constraints**: The size and category of the training data set must not be too large
- **SVM Parameters:** Good kernel along with a chosen value of 'C' and 'Gamma' must be used with SVM while implementation.
- **Color Gradient of Image:** The image taken must be such that on reducing the image to grayscale, a lot of it's features are not getting hidden.
- **Size of the Image:** The size of all the images in pixels must either be or must be reduced to ta particular value in implementation.

## 1.2 Basics

### 1.2.1 Computer Vision

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and in general, deal with the extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory

Applications range from tasks such as industrial machine vision systems which, say, inspect bottles speeding by on a production line, to research into artificial intelligence and computers or robots that can comprehend the world around them. The computer vision and machine vision fields have significant overlap. Computer vision covers the core technology of automated image analysis which is used in many fields. Machine vision usually refers to a process of combining automated image analysis with other methods and technologies to provide automated inspection and robot guidance in industrial applications. In many computer vision applications, the computers are pre-programmed to solve a particular task, but methods based on learning are now becoming increasingly common. Examples of applications of computer vision include systems for:

- Controlling processes, e.g., an industrial robot;
- Navigation, e.g., by an autonomous vehicle or mobile robot;
- Detecting events, e.g., for visual surveillance or people counting;
- Organizing information, e.g., for indexing databases of images and image sequences;
- Modeling objects or environments, e.g., medical image analysis or topographical modeling;

- Interaction, e.g., as the input to a device for computer-human interaction, and
- Automatic inspection, e.g., in manufacturing applications.

Typical functions which are found in many computer vision systems are:

- Image Acquisition
- Pre-processing
- Detection / Segmentation
- High-level Processing
- Decision Making

### 1.2.2 Machine Learning

Machine learning is the subfield of computer science that "gives computers the ability to learn without being explicitly programmed". Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data– such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is unfeasible; example applications include spam filtering, optical character recognition (OCR), search engines and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses in prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning.

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are:

- Supervised Learning
- Unsupervised Learning

Some of the common approaches of Machine Learning are:

- Support Vector Machines(SVM)
- Decision Tree Learning
- Artificial Neural Networks
- Deep Learning

## 1.3 Algorithms Used in Project

### 1.3.1 Support Vector Machines

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

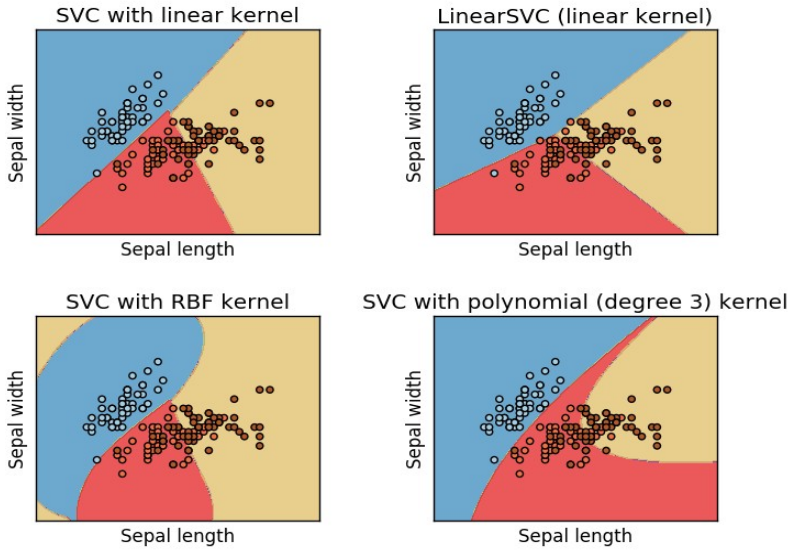


**Figure 1.1: Classification using different SVM Kernels**

### 1.3.2 K-Nearest Neighbour

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. In both cases, the input consists of

the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

#### *1.3.2.1 Feature Extraction*:

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction.

In computer vision, the feature extraction and dimension reduction pre-processing is done by algorithms such as :

- Haar face detection
- Mean-shift tracking analysis
- PCA or Fisher LDA projection into feature space, followed by k-NN classification

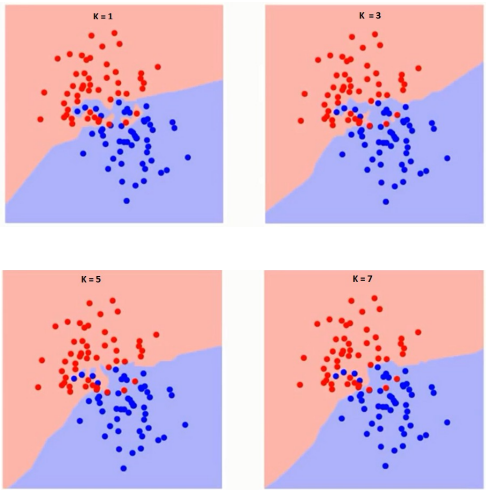


**Figure 1.2 : Classification with different values of K**

### 1.3.2 : PCA(Principal Component Analysis)

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

Some of the generalizations of the PCA are:

- Nonlinear generalizations
- Multilinear generalizations
- Robustness – weighted PCA
- Robust PCA via decomposition in low-rank and sparse matrice
- Sparse PCA

#### *1.3.2.1 Limitations of PCA*

Some of the limitation of PCA are:

- The results of PCA depend on the scaling of the variables.
- The applicability of PCA is limited by certain assumptions made in its derivation.

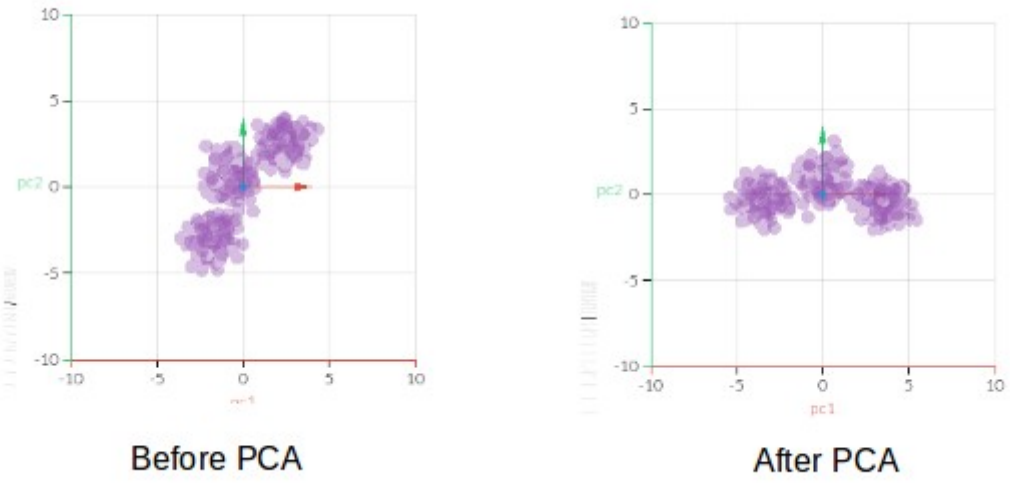


**Figure 1.3: Dimensionaltiy Reduction using PCA**

## 1.4 Introduction to Python and Libraries Used

### 1.4.1 Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site,https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications

Features of Python include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

### 1.4.2 Scikit Learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012. As of 2015, scikit-learn is under active development and is sponsored by INRIA, Telecom ParisTech and occasionally Google (through the Google Summer of Code).

Salient Features :-

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classes imported from SkLearn:

- SVC - for SVM
- GridSearchCV – Tuning the hyper-parameters of an estimator
- Neighbours – for KNN

### 1.4.3 matplotlib

matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB. SciPy makes use of matplotlib.

matplotlib was originally written by John D. Hunter, has an active development community, [1] and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in 2012.

As of 30 October 2015, matplotlib 1.5.x supports Python versions 2.7 through 3.5. Matplotlib 1.2 is the first version of matplotlib to support Python 3.x. Matplotlib 1.4 is the last version of matplotlib to support Python 2.6

# Chapter 2

# Literature Survey

## 2.1 Literature Survey

### 2.1.1 Abstract of the Papers Referred

K-Nearest Neighbor algorithm(kNN) and Support Vector Machines(SVM) both have their own advantages and disadvantages. While NN classifiers can deal with the hugely multiclass nature of visual object recognition effortlessly, they suffer from the problem of high variance in the case of limited sampling. Alternatively one can use SVMs as they provide impressive performance on high-dimensional data but they involve time-consuming optimization and computation of pairwise distances.

Our chosen framework on the other hand is a hybrid of these two methods which deals naturally with the multiclass setting, has reasonable complexity both in training and rum time, and yields excellent results in practice. The basic idea is to find close neighbours to a query sample and train a local support vector machine that preserves the distance function on the collection of neighbors.

This method can be applied to large, multiclass data sets for which it outperforms nearest neighbor and support vector machines, and remains efficient when the problem becomes intractable for support vector machines. We will use the Caltech-101 dataset to train our model and for classifying categories.

### 2.1.2. Findings from the Research Papers

As per the research papers, object recognition can be done perfectly by using Neural Networks and Deep Learning over the conventional algorithms like SVM and KVM. As per the current status, multi-layer neural networks are the best suited option for object recognition. Moreover, convolutional neural networks are still used in today's classifiers.

One of the research papers also state that the combination of both the algorithms i.e SVM and KNN and lead to improvements. SVMs can work as good as CNNs provided good features are used with a good kernel function.

### 2.1.3 Problem Statement:-

Given two sets one of training data and other of validation data consisting of images, you must train your classifier on the training data so as to determine the class/category of object present in one of the images from the validation data. You are also required to optimize the time and space needed for the given validation data to train your classifier.

Calculate the time taken by each processes e.g. PCA, KNN SVM. Time should be noted in seconds and percentage of accuracy can be in point or in percentage. Make sure increase in number of training examples should be directly proportional to increase in accuracy.

## 2.1.4 Solution Proposed

### *2.1.4.1 List of objectives to be achieved:*

The following objectives achieved by the proposed solution:

- The traning of the classifer on the validation data is done on a large set of training data, which is intractable by using naive algorithms.
- The memory used to train the classifier is reduced so that it doesn't affect the other programs running on the same system
- The category of the object for the given image is determined with reduced amount of time as compared to naive algorithms for object categorization and image recognition.
- Accuracy of the classifier is improved as compared to naive Bayes Classifier, or just the use of SVM or KNN alone.

### *2.1.4.2 Methodology to achieve the above listed objectives*

To achieve the above listed objectives we train a support vector machine(SVM) on the collection of nearest neighbors.

A naive version of the SVM-KNN is: for a query,

1. compute distances of the query to all training examples and pick the nearest K neighbors;
2. if the K neighbors have all the same labels, the query is labeled and exit; else, compute the pairwise distances between the K neighbors;
3. convert the distance matrix to a kernel matrix and apply multiclass SVM;
4. use the resulting classifier to label the query.

Three variants of multiclass SMV produce roughly the same quality of classifiers and the DAGSVM is chosen for its better speed.

The naive version of SVM-KNN is slow mainly because it has to compute the distances of the query to all training examples. Here we again borrow the insight from psychophysics that humans can perform fast pruning of visual object categories. In our setting, this translates into the practice of computing a “crude” distance (e.g. L 2 distance) to prune the list of neighbors before the more costly “accurate” distance computation. The reason is simply that if the crude distance is big enough then it is almost certain that the accurate distance will not be small. This idea works well in the sense that the performance of the classifier is often unaffected whereas the computation is orders-of-magnitude faster. We term this idea “shortlisting”.

An additional trick to speed up the algorithm is to cache the pairwise distance matrix in step 2. This follows from the observation that those training examples who

participate in the SVM classification lie closely to the decision boundary and are likely to be invoked repeatedly during query time.

After the preceding ideas are incorporated, the steps of the SVM-KNN are: for a query,

1. Find a collection of K sl neighbors using a crude distance function (e.g. L 2 );
2. Compute the “accurate” distance function (e.g. tangent distance) on the K sl samples and pick the K nearest neighbors
3. Compute (or read from cache if possible) the pairwise “accurate” distance of the union of the K neighbors and the query
4. Convert the pairwise distance matrix into a kernel matrix
5. Apply DAGSVM on the kernel matrix and label the query using the resulting classifier.

# Chapter - 3

# Implementation

## 3.1 Environment Setup

### 3.1.1 Download and Install Conda

Conda is a package manager application that quickly installs, runs, and updates packages and their dependencies. The conda command is the primary interface for managing installations of various packages. It can query and search the package index and current installation, create new environments, and install and update packages into existing conda environments.

A conda package is a compressed tarball file that contains system-level libraries, Python or other modules, executable programs, or other components. Conda keeps track of the dependencies between packages and platforms.

To download conda, use the command as :

```
wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh
```

This will download the bash file for conda, to install you must download the bash file as:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

Once installed you can list the conda files using the command :

```
conda install
```

### Install Anaconda

Anaconda ("Anaconda Distribution") is a free, easy-to-install package manager, environment manager, Python distribution, and collection of over 720 open source packages with free community support. Hundreds more open source packages and their dependencies can be installed with a simple “conda install [packagename]”. It’s platform-agnostic, can be used on Windows, OS X and Linux.

Since conda is already installed, we an make the use of it to install the anaconda, using the command :

```
conda install anaconda
```

This will install all the libraries and packages needed to run the project.

Anaconda Navigator is a desktop graphical user interface included in Anaconda that allows you to launch applications and easily manage conda packages, environments and channels without the need to use a command prompt or terminal program. Anaconda Navigator is automatically installed when you install Anaconda.

Packages available in Anaconda:

- Over 150 packages are automatically installed with Anaconda.
- Over 250 additional open source packages can be individually installed from the Anaconda repository at the command line, by using the “conda install” command.

## 3.2 Starting Server

All the other interfaces —- the Notebook, the Qt console, ipython console in the terminal, and third party interfaces —- use the IPython Kernel. The IPython Kernel is a separate process which is responsible for running user code, and things like computing possible completions. Frontends, like the notebook or the Qt console, communicate with the IPython Kernel using JSON messages sent over ZeroMQ sockets; the protocol used between the frontends and the IPython Kernel is described in Messaging in Jupyter

A kernel process can be connected to more than one frontend simultaneously. In this case, the different frontends will have access to the same variables.

This design was intended to allow easy development of different frontends based on the same kernel, but it also made it possible to support new languages in the same frontends, by developing kernels in those languages, and we are refining IPython to make that more practical.

In oder to start the execution of the code, we use Jupyter-notebook as our environment. Jupyter-notebook start and executes the code snippet in its kernel. In order to start to jupyter notebook, go to the directory to which the file is present from the terminal and enter the below command.

```
jupyter-notebook
```

The command will start the jupyter server at the localhost. The port number used is 8888. The jupyter-notebook can be accessed from browser now with url as localhost:8888

The output of the above command will as:

```
nikhil@nikhil-Vostro-1320 ~/Desktop/Academics/Minor Project/Minor-Project $ jupy
ter-notebook
[W 04:57:30.936 NotebookApp] Unrecognized JSON config file version, assuming ver
sion 1
[I 04:57:37.206 NotebookApp] [nb_conda_kernels] enabled, 1 kernels found
[I 04:57:37.254 NotebookApp] Writing notebook server cookie secret to /run/user/
1000/jupyter/notebook_cookie_secret
[I 04:57:38.296 NotebookApp] ✓ nbpresent HTML export ENABLED
[W 04:57:38.296 NotebookApp] ✗ nbpresent PDF export DISABLED: No module named nb
browserpdf.exporters.pdf
[I 04:57:38.308 NotebookApp] [nb_conda] enabled
[I 04:57:38.833 NotebookApp] [nb_anacondacloud] enabled
[I 04:57:38.862 NotebookApp] Serving notebooks from local directory: /home/nikhi
l/Desktop/Academics/Minor Project/Minor-Project
[I 04:57:38.862 NotebookApp] 0 active kernels
[I 04:57:38.862 NotebookApp] The Jupyter Notebook is running at: http://localhos
t:8888/
[I 04:57:38.863 NotebookApp] Use Control-C to stop this server and shut down all
 kernels (twice to skip confirmation).
```

**Figure 3.1 : Starting the server**

## 3.3 Implementation Process

### 3.3.1 Importing Libraries

The first step in the implmentation is to import the libraries needed for running the process. In one cell of the jupyter-notebook we import all the libraries required for the project, this ensures that if independent cells are run while executing the rest of the code, there is no error due to missing libraries to the function and the modularity of the functions can be maintained easily. We mainly use scikit-learn as our library so we will import the library functions needed for both SVM and KNN.

```
import os
import itertools
from time import time

import matplotlib
import matplotlib.pyplot as plt
import numpy as np

from skimage import io
from sklearn.neighbors import NearestNeighbors
from sklearn.svm import SVC
from skimage.transform import resize
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

%matplotlib inline
```

**Figure 3.2 : Importing libraries**

### 3.3.2 Reading the Dataset

In the next step we will get the Caltech 101 dataset as input for our program. After downloading the caltech dataset we will have to unzip the dataset in order to read the images. For reading the data we will iterate over all the folders in present in the dataset. Since we know that only the heirarchy of the directory is just single level we don't need any recursive procedure call to get all images as our input we can do it by just having a single loop of iteration.

While we are reading the input from the folder we divide the dataset into two parts namely training and validation. The use of training dataset is to train the classifier, and the validation dataset provides the values for checking the accuracy and efficiency of the classifier. We can use any number of images from the dataset for each type of dataset keeping in mind the total number of images in each type of dataset. While we read the data, we also store the label for each of the figure that is why we have to create a object to use key-value pair with array.

### 3.3.3 Prepare training data images

We now have the data ready for processing, so will now try to process the images by trying to flatten the image by storing the pixels of the form 2-D to the 1-D for each of the images which will lead to training image matrix. In this step we will also seperate the flattened trainng matrix and training labels from the object.

### 3.3.4 Apply PCA

We now apply PCA to the image matrix from previous step to reduce the nuimber of dimensions for each image. This will reduce the number of dimensions of the image which are needed for processing. This decrease will help to reduce time needed to run the SVM on the given set of images.

PCA is used to decompose a multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance. In scikit-learn,**PCA**is implemented as a *transformer* object that learn components in its`fit`method, and can be used on new data to project it on these components.

The optional parameter `whiten=True` makes it possible to project the data onto the singular space while scaling each component to unit variance. This is often useful if the models down- stream make strong assumptions on the isotropy of the signal: this is for example the case for Support Vector Machines with the RBF kernel and the K-Means clustering algorithm.

### 3.3.5 Eigen vector on orthonormal facees

Throughout, we work in the Euclidean vector space V = R n, the space of column vectors with n real entries. As inner product, we will only use the dot product v·w = v T w and corresponding Euclidean norm k v k = √ v · v . Two vectors v, w ∈ V are called orthogonal if their inner product vanishes: v · w = 0. In the case of vectors in Euclidean space, orthogonality under the dot product means that they meet at a right angle. A particularly important configuration is when V admits a basis consisting of mutually orthogonal elements.

What are the advantages of orthogonal and orthonormal bases? Once one has a basis of a vector space, a key issue is how to express other elements as linear combinations of the basis elements — that is, to find their coordinates in the prescribed basis. In general, this is not so easy, since it requires solving a system of linear equations. In high dimensional situations arising in applications, computing the solution may require a considerable, if not infeasible amount of time and effort.

However, if the basis is orthogonal, or, even better, orthonormal, then the change of basis computation requires almost no work.This is the crucial insight underlying the efficacy of both discrete and continuous Fourier analysis, least squares approximations and statistical analysis of large data sets, signal, image and video processing, and a multitude of other applications.

### 3.3.6 KNN

In KNN we have k as the number of nearest neighbours as our input and the processed image for which we want to predict the label. This will also use the matrix obtained after the projection of the input values on orthonormal faces.

We will fit the NearestNeighbors for plotting the KNN on the projection matrix and will try to find the label by using processed image as the input to the fitted variable. In our project this function is just a utility function for the complete process, the complete process is yet more complex than this. This is the just the initial processing to reduce the number of dataset on which the final SVM will run.

### 3.3.7 Checking Similarity

Once the indices of the nearest neighbours are obtained from the KNN, we have the check the similarity of the nearest neighbours predicted in the previous proces. If all the predictions are the same, this clearly states that the prediction is of only one of the labels and there is no ambiguity or desparity. Hence we can conclude that the correct label of the image can be predicted by just using the KNN.

If the predicted neighbours are not the same, there is still a doubt about the class of the image which can be predicted by training SVM classifier. SVM classifier once trained will find the most probable label for the given image.

### 3.3.8 SVM

In SVM function we will first find the best values of C and gamma to the classifier to be trained which will be used as a parameters for training the classifier. In this step we are using “rbf” as our kernel. The rbf kernel is one of the kernel defined by SVM, so we don't need to build our own kernel.

We have already found the nearest neighbours to the given image and we need to just find the actual label of the image. For this we wil take the nearest neighbours as the input to the SVM classifier and then train our SVM classifier on this set. This gurantees a significant decrease in the number of values on which the classifer is to be trained. Once the classifier is trained we predict the label of the selected image.

In order to increase the performace of the image prediction we have to ensure that the value of k selected in the KNN is upto the requirements so that the dataset not always give same nearest neighbours as well as the value of k must be less so that SVM has to be trained on considerabley lesser number of images. If the value of k is extremely small, the prediction will always be made with the help of only the KNN, and if the value of k is extremely large the label will always be found with the help of both KNN and SVM. But taking very large value of k is useless, this is similar to finding the label of the image by just using SVM, the part of the knn used will be waste. In order to get better results the SVM classifier can be trained on the kernel made by dertermining the approximate distance instead of actual one.

### 3.3.9 Accuracy

In order to determine the accuracy of the classifier we must use the validation data from the given dataset. Validation images are defined by their classses but the same image is not used for training, so the classifier will try identify the image based on the characteristics it percieved from the trianing examples and the characteristics of the current image.

In order to find the accuracy of the classifier, we will process and add the image from each class to our validation matrix and then will iterate on all of the m to allow the classifier to find the label / class of the image.

The accuracy is the division of number of images for which correct label are found per each class and the total number of validation images used.

# Chapter - 4

# Results and  Screenshots

## 4.1 Dataset Input

### 4.1.1 Categories of Images

After we open the folder, the number of categories can be seen, which sums up a total of 101 category of the image. The categories are shown as :

```
['sunflower', 'gramophone', 'wheelchair', 'trilobite', 'sea_horse', 'ferry', 'water_lilly', 'chair', 'joshua_tree',
 'crab', 'rooster', 'emu', 'starfish', 'mayfly', 'yin_yang', 'airplanes', 'grand_piano', 'llama', 'scissors', 'garf
ield', 'lamp', 'cannon', 'lobster', 'stegosaurus', 'cougar_body', 'watch', 'Faces', 'electric_guitar', 'binocular',
 'accordion', 'stop_sign', 'pigeon', 'ant', 'pagoda', 'metronome', 'car_side', 'anchor', 'menorah', 'stapler', 'umb
rella', 'Faces_easy', 'cellphone', 'headphone', 'wrench', 'scorpion', 'pizza', 'Motorbikes', 'nautilus', 'chandelie
r', 'elephant', 'rhino', 'bass', 'crocodile_head', 'BACKGROUND_Google', 'hawksbill', 'barrel', 'crayfish', 'dolphi
n', 'platypus', 'brain', 'lotus', 'flamingo', 'ceiling_fan', 'beaver', 'panda', 'ketch', 'bonsai', 'strawberry', 'h
edgehog', 'tick', 'flamingo_head', 'Leopards', 'euphonium', 'gerenuk', 'dollar_bill', 'windsor_chair', 'laptop', 's
axophone', 'butterfly', 'soccer_ball', 'dalmatian', 'brontosaurus', 'mandolin', 'ibis', 'wild_cat', 'snoopy', 'cu
p', 'minaret', 'camera', 'octopus', 'cougar_face', 'schooner', 'inline_skate', 'revolver', 'crocodile', 'helicopte
r', 'dragonfly', 'ewer', 'kangaroo', 'pyramid', 'okapi', 'buddha']
```

**Figure 4.1 : Categories of images present**

### 4.1.2 The Object of Dataset

As mentioned previously the object is composed of dictionay and list in python and looks like json. A screenshot of the object by jsonviewer is shown below:

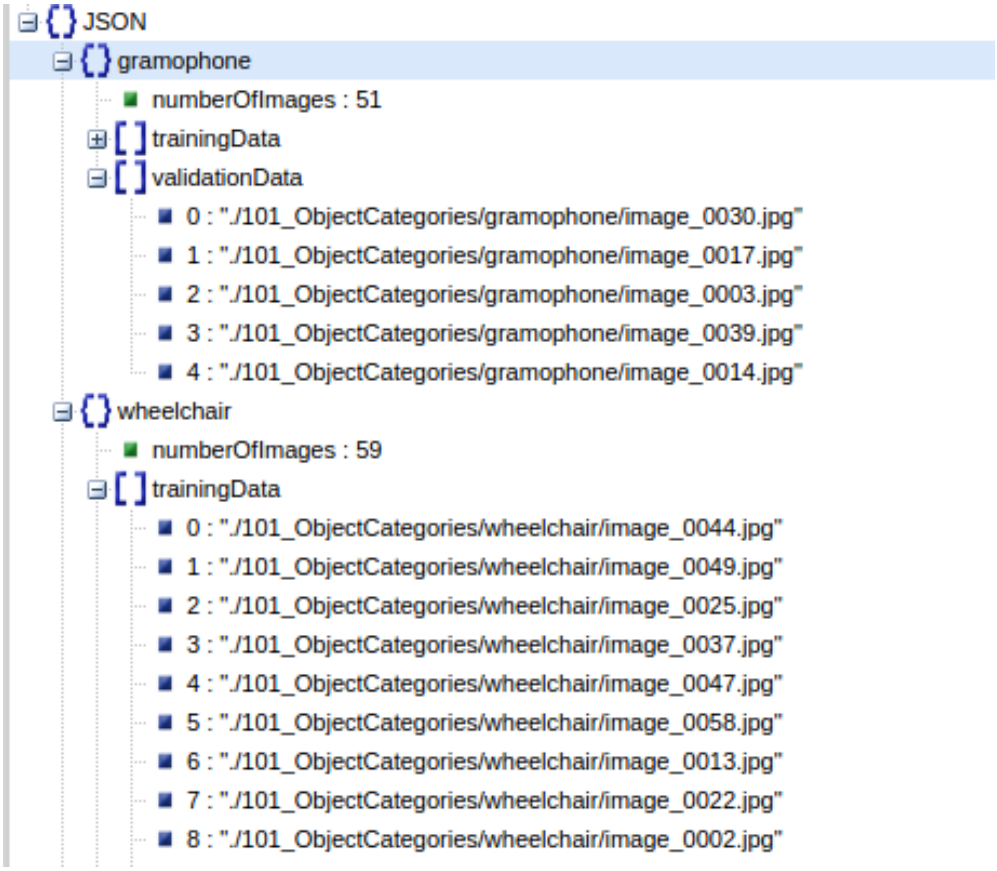


**Figure 4.2 : JSON view of the prepared object**

## 4.2 Image Processing

### 4.2.1 Flatten Image

In this process we take the each pixel of the image and convert them to the linear array. So the final trainingMatrix will comprise of the information about each image in the row and each row contains the pixels of that image. The output of this will look something like this, though the show output is shortened to get the glimpse of the output

```
for i in trainingMatrix:
    print i[:10]
```

```
[0.34919051447692534, 0.32351562895385461, 0.31423078687331035, 0.30782007887753721, 0.28742334293764932, 0.2771555
2563108914, 0.27137520392156755, 0.27137520392156755, 0.27494383137254796, 0.26780657647058714]
[0.3514928589878959, 0.34277091464129561, 0.33952862745097101, 0.33952862745097107, 0.33891808383232541, 0.33070539
15697924, 0.32859725490195196, 0.32859725490195207, 0.32270352941175623, 0.32270352941175634]
[0.16282744962623386, 0.14785029111971787, 0.14294630457515969, 0.13777914869085039, 0.13221082896950839, 0.1328636
7455676554, 0.13170216078431038, 0.13170216078431038, 0.13170216078431041, 0.13217528758169603]
[0.95556862745095328, 0.95949019607840391, 0.95949019607840369, 0.96341176470585421, 0.95724762240222305, 0.9479250
9099445399, 0.9547350005870332, 0.9615449101796123, 0.96341176470585288, 0.95559210989782084]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.49043144534456423, 0.49679909827399804, 0.40883305741455073, 0.37671063872252164, 0.39318000939294573, 0.3846834
7775034856, 0.39994828695546641, 0.41370547375833011, 0.41792528355051017, 0.40879778090873614]
[0.40353594370082063, 0.40626871345542903, 0.41583340759655835, 0.41932642156861399, 0.42876783286367093, 0.4472602
5977455447, 0.45828380092754234, 0.4672731751203329, 0.48558067776210778, 0.49527026153573672]
[0.39314403898086475, 0.4704715674533374, 0.59746572971704515, 0.53359042385815281, 0.3598361723611489, 0.285146065
51602639, 0.28117495831865663, 0.34818307854879321, 0.36303366208758664, 0.33582693436656053]
[0.095291653888300382, 0.12143567956244278, 0.10313715895659616, 0.094716886213845872, 0.11510427886188233, 0.09850
3364737191845, 0.10073180980783386, 0.13034640894289715, 0.0947686355328555, 0.11432302737270485]
[0.13974553368557366, 0.12740910849673442, 0.13062182032797365, 0.13713459869281305, 0.12821471297405523, 0.1266326
2225353392, 0.13885257450589189, 0.14686008888889165, 0.12965121252397499, 0.21770722367030651]
[0.099515954901958345, 0.099752047058821097, 0.099752047058821097, 0.099752047058821111, 0.10404776078431145, 0.107
30714901960532, 0.11203030196078174, 0.11472638039215413, 0.12052049803921293, 0.11028974705882091]
[0.31801279558526813, 0.29023184454616857, 0.30644027122222234, 0.34712987671711076, 0.3568869437595078, 0.36419185
159088657, 0.41730080075138687, 0.42744657625919741, 0.43880732300100433, 0.45132812492656615]
```

**Figure 4.3 : Code and Output after flattening the image**

### 4.2.2 PCA

After flattening we apply PCA so the processed data now looks like this, which is pretty less than previous image data

```
PCA Started
Time taken by PCA 0.334s::

array([  3.63981826e-01,   1.10923168e-01,   8.67959014e-02,
         6.32381236e-02,   5.22770495e-02,   4.62504677e-02,
         4.20161613e-02,   3.61640022e-02,   3.26179238e-02,
         2.70447645e-02,   2.36386750e-02,   2.12808018e-02,
         1.83468232e-02,   1.66755632e-02,   1.49461479e-02,
         1.38530273e-02,   1.13877252e-02,   9.87363084e-03,
         8.68821736e-03,   1.30387034e-31])
```

**Figure 4.4 : Output of PCA**

## 4.3 Prediction

Once the process of KNN and SVM is complete we get the output of the image given for the prediction from the validation data. The output looks like

```
make_prediction(5)
```

```
Predicted class is mentioned below:
sunflower


Chosen image is displayed below:
```


**Figure 4.5 : Prediction of the image from validation data**

## 4.4 Accuracy

Since the clasisfier is trained by the combination of two algorithms, in order to find the accuracy of the classifier we have to write our own custom accuracy method which gives the output of the classifier as shown below

```
---Finding accuracy---
Total time taken = 3690.125891
Total number of correct predictions are 40
Accuracy =  0.28
```

**Figure 4.6 : Output showing accuracy of the classifier**

# Chapter-5

# Future Scope and Conclusion

## 5.1 Future Scope

Most significant scope of this hybrid implementation can be seen in the area of Computer Vision. As we have already mentioned that human eyes can recognize 30,000 distinct categories and that also in limited timeframe while current state of the art techniques of Computer Vision can only recognize 100 to 150 distinct categories and hence this implemntation is a step forward to make computer systems reach human eye level accuracy.

This work can further be improved by embedding Artificial Neural Network(ANN) to further improve the accuracy. ANN can be used in place of kNN to filter most significant results to train SVM clssifier.

## 5.2 Conclusion

In this implementation we have shown that SVM classifier can be used for multiclass classification even on high dimensional data. To ensure that problem don't become intractable on large number of training examples limited objects are used to construct the decision boudary and selection of those examples is made via kNN technique. Using best of both worlds we were able to improve the accuracy significantly.

# Chapter – 6

# References

## 6.1 References

[1] Hao Zhang , Alexander C. Berg, Michael Maire, Jitendra Malik. *SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition*. Computer Science Division, EECS Department Univ. of California, Berkeley, CA 94720

[2]Le Hoang Thai, Tran Son Hai, Nguyen Thanh Thuy. *Classification using Support Vector Machine and Artificial Neural Network*. I.J. Information Technology and Computer Science, 2012, 5, 32-38

[3]Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di , Yizhou Yu . *HD-CNN: Hierarchical Deep Convolutional Neural Network for Large Scale Visual Recognition*. University of Illinois at Urbana-Champaign, Carnegie Mellon University eBay Research Lab, The University of Hong Kong

[4]Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla & Antonio Criminisi. *Training CNNs with Low-Rank Filters for efficient Image Classification.* University of Cambridge, Microsoft Research

[5]Bernard Haasdonk(1) and Claus Bahlmann(2). *Learning with Distance Substitution Kernels.* (1)Computer Science Department Albert-Ludwigs-University Freiburg 79110 Freiburg, Germany, (2)Siemens Corporate Research, Inc. 755 College Road East Princeton, NJ 08540, USA

[6]Kaiming He , Xiangyu Zhang , Shaoqing Ren , Jian Sun. *Deep Residual Learning for Image Recognition.* Micosoft Research