Image Classification Using Traditional Machine Learning

Madhura Zambare
(11001)
Madhura Deshpande
(11005)

INDEX

Abstract

Classifying different images into certain categories is a simple task for any human being, but it can be a very complex problem for a machine to solve. Image classification using Artificial Intelligence is a very crucial problem as it has many applications in the real world. This project uses a traditional machine learning approach to answer this problem. Traditional ML classifiers are easy to understand and implement, fast and provide good explanations of the data and predictions. However, their performance is compromised when the data is high dimensional and large. We have analysed and compared ML algorithms, namely K Nearest Neighbours, Naïve Bayes, Random Forest and Support Vector Machine to classify different yoga pose images into appropriate asanas. We tried to improve the results by techniques like feature extraction and edge detection by using Gabor filters and Sobel kernels. The project describes the limitations of traditional ML classifiers and suggests the need for deep learning techniques to handle this problem more accurately.

***

Objectives

1. Feature Extraction from Images.
2. Classify images using traditional machine learning techniques.
3. Analyse and compare the performance of different machine learning algorithms.
4. Determine the best machine learning algorithm for the yoga pose dataset.

***

Motivation

Looking at different images and classifying them is a fairly simple task for a human being, but it is very complicated for a machine to perform the same task. We wanted to know how computers process image data and learn from it using different feature extraction methods. Further, we wanted to check the performance of different machine learning techniques for image classification. We have used only traditional machine learning algorithms to investigate whether these algorithms can perform as good as deep learning algorithms. We were looking for the advantages and limitations of various classification models by comparing them based on their performance.

***

Data Description

Data source: https://www.kaggle.com/niharika41298/yoga-poses-dataset

Yoga pose data is secondary data taken from Kaggle. It contains images of the 5 most famous asanas namely downward dog, plank pose, tree pose, goddess pose and warrior-2 pose in jpg format. The main goal of the dataset is to create a yoga pose image classifier for the asanas. The images are scraped directly from the web so some text/captions may be present in some images but they won't hinder training.

All Images are coloured each with different sizes. There is a total of 1550 images. The data is already divided into train and test sub-directories which is further unequally distributed among 5 directories. This means that the dataset is imbalanced.

Pose	Train images	Test images	Total images
Downdog	223	97	320
Goddess	180	80	260
Plank	266	115	381
Tree	160	69	229
Warrior-2	251	109	360

Images :

1. Downdog

2. Goddess

3. Plank

4. Tree

5. Warrior

The reason we chose this dataset is that some yoga poses are very similar to each other. For example, observe the goddess pose and the warrior pose in the above images. The positions of the legs, hands and torso are very similar. This makes the classification of yoga poses a difficult task for a machine.

***

Libraries

Tool: Python

Libraries:

- Numpy:
  Used for scientific calculations on arrays. All the images with their labels are stored in the form of NumPy arrays. These arrays are fed to the machine learning models as inputs.

- Pandas:
  Data manipulation of data frames.

- Glob:
  Used to return all file paths that match a specific pattern.

- Open-cv:
  Image processing and feature extraction.

- OS:
  Provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

- Sci-kit learn:
  Building machine learning models and analysing their performance.

- Skimage:
  Used for feature extraction from images.

- Matplotlib and Seaborn:
  Data visualization – To plot heatmap and bar graph to compare accuracies of different ML models.

***

Data Pre-processing

- Read an Image –
  In this step, we simply store the path to our image dataset into a variable and then we create a function to load folders containing images into arrays so that computers can deal with them. OpenCV has a method cv2.imread for converting images into NumPy arrays.

- Resize an Image –
  The images in the dataset vary in size, therefore, we should establish a base size for all images fed into our ML algorithms by resizing them. We have resized all the images into 32 x 32 dimensions.

- Grey-scaling of an Image –
  Since the classification of yoga poses does not depend on the colours of the images, we have used grey-scaling to read the images. This reduces the dimensions of each image as the grey-scaled image has only one channel giving array dimensions (32,32) while RGB images have 3 channels giving array dimensions (32,32,3). The image will be converted to a Grey-scale (range of Gray shades from white to black) the computer will assign each pixel a value based on how dark it is.

Original Image

Grey-scale Image

- Normalization –
  The pixel values in the image range from 0 to 255. Because of this wide range of pixels, we need to normalize the pixel arrays so that all the values are between 0 and 1, thus bringing all the values of numeric arrays in the dataset to a common scale.

- Feature Extraction –
  Features are the marked properties that are unique and help us identify the particular object, image, or anything. While working on an image dataset we need to extract the features of different images which will help us segregate the images based on certain features or aspects. We have used the following 2 methods for feature extraction –

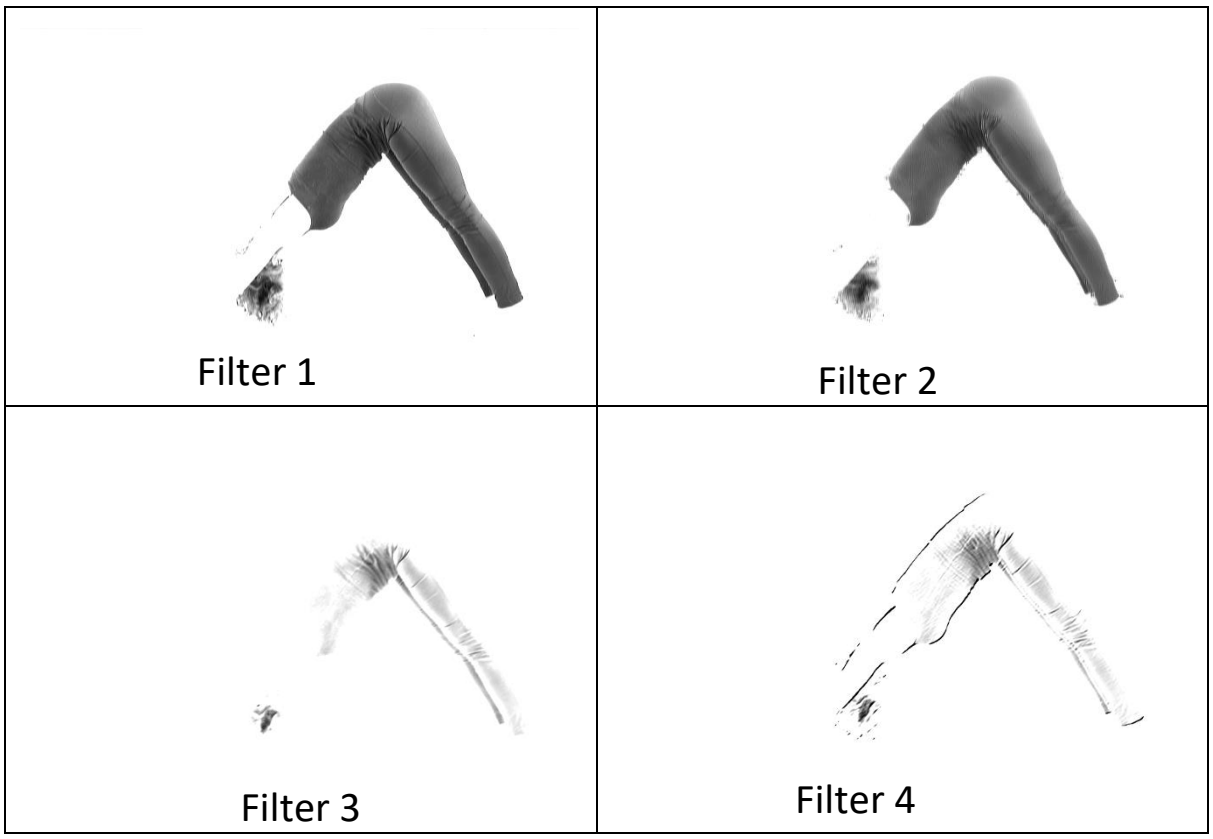
  A. Gabor Filters: The Gabor filter is a linear filter used widely in image processing applications for edge detection, texture analysis, feature extraction, etc. Gabor filters are special classes of bandpass filters, i.e., they allow a certain 'band' of frequencies and reject the others. When a Gabor filter is applied to an image, it gives the highest response at edges and at points where texture changes. Certain parameters control how the Gabor filter will be and which features will it respond to. The equation is given as:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right)$$

  where,
  λ — Wavelength of the sinusoidal component.
  Θ — The orientation of the normal to the parallel stripes of the Gabor function.
  Ψ — The phase offset of the sinusoidal function.
  σ — The sigma/standard deviation of the Gaussian envelope

ɣ — The spatial aspect ratio and specifies the ellipticity of the support of the Gabor function.

We have applied 4 Gabor filters using the OpenCV package to each image by changing the values of Θ and σ and the results are as follow -



B. Edge Detection: Edges in an image are the corners where the pixel changes drastically, as the images are stored in array form, we can visualize different values and see where the change in pixel value is higher. We have used Sobel Kernel for edge detection. When using Sobel Edge Detection, the image is processed in the X and Y directions separately by kernel

convolution, and then combined to form a new image that represents the sum of the X and Y edges of the image. The Kernels used in this process are given below :

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

Results of Edge Detection

- Label Encoding - Encode the labels of the images (downward dog, plank pose, tree pose, goddess pose and warrior) to integers (0 - 4) and store them into different arrays for the ease of computation. We have used the Label Encoder method in the sci-kit learn package for this.

***

Model Building

We used the following Supervised machine learning algorithms for image classification:

- K-Nearest Neighbour :
  K-NN algorithm stores all the available data and classifies a new data point based on the similarity. The nearest neighbours are those data points that have minimum distance in feature space from our new data point. And K is the number of such data points we consider in our implementation of the algorithm. Therefore, distance metric and K value are two important considerations while using the KNN algorithm.
  - Parameters:
    KNeighborsClassifier(metric='manhattan',n_neighbors=4)
  - Accuracy: 56%

- Naïve Bayes Classifier :
  Naïve Bayes algorithm is based on Bayes theorem (conditional probability). It is a probabilistic classifier which means it predicts on the basis of the probability of an object.
  - Parameters:
    GaussianNB(priors=None, var_smoothing=5e-05)
  - Accuracy:32%

- Random Forest :
  Random Forest is a classifier that contains many decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset

● Parameters :

```
RandomForestClassifier(n_estimators = 100,
                       random_state = 11,
                       max_depth = 7,
                       min_samples_leaf= 20)
```

● Accuracy: 71%

➢ Support Vector Machine (SVM) :

SVM algorithm is used to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

● Parameters:

```
svm.SVC(kernel = 'rbf', gamma = 0.00008, C = 1,
        decision_function_shape= 'ovr',
        random_state = 0)
```

● Accuracy: 66%

***

Comparison

- Since images of goddess and warrior poses are very similar to each other K-NN model misclassifies the poses in the warrior pose as goddess pose and that's why the K-NN algorithm gives moderate accuracy i.e., 56%. The reason for this could be the high dimensionality of the image dataset.

- The Naïve Bayes classifier is not complex enough to learn from high dimensional datasets like images. It performs poorly with an accuracy of 33%. The majority of images are misclassified into plank pose. This classifier assumes the independence between all features, which may not be the case every time.

- Random Forest is based on the concept of ensemble learning and hence it can handle complex datasets. Random forest is great with high dimensional data since it works with subsets of data (thus requires less time). Random Forest is indifferent to non-linear features, handles outliers and unbalanced datasets.

  The random forest classifier performs the best among all the models giving overall accuracy of 71%. But it struggles to classify the poses in goddess pose giving a lot of mis-classifications in warrior pose.

- The yoga poses dataset is a non-linear complex dataset and SVM can learn from it because of the kernel trick, which is the real strength of SVM. It scales relatively well to high dimensional data. Hence SVM model with non-linear kernel RBF performs well giving an accuracy of 68%. The mis-classification problem in goddess pose and warrior pose still exists. This classifier takes the most time to train among all the others.

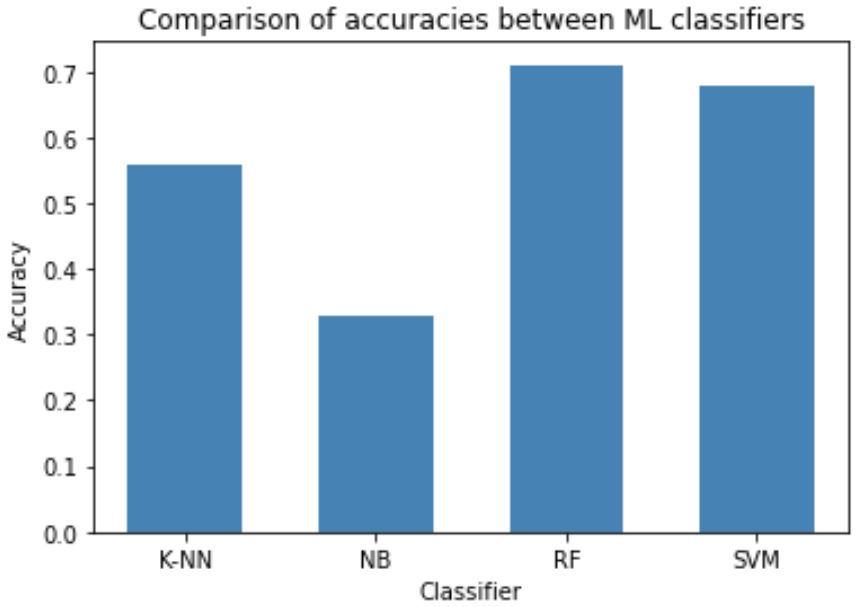
The following table gives a comparison of accuracy, precision, recall and F1-score between machine learning models :

Algorithm	**Testing Accuracy**	**Weighted Precision**	**Weighted Recall**	**Weighted F1-score**
K-NN	0.56	0.57	0.56	0.56
Naïve Bayes	0.32	0.45	0.32	0.31
Random Forest	0.71	0.75	0.70	0.68
SVM	0.66	0.7	0.67	0.66

***

Conclusion

This project tackles the problem of image classification using traditional machine learning techniques. We have used K-Nearest Neighbours, Naïve Bayes, Random Forest and Support Vector Machine and the bar graph below compares their performance.



Random Forest is performing the best among all other models because of its ability to work with high dimensional data. SVM performs well too with the help of kernel trick. But in comparison to the average performance of deep learning algorithms such as Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN), traditional machine learning algorithms are still far behind.

Machine learning works well on smaller data set with a limited number of dimensions, while the performance of deep learning increases as data increases. This is why we observed significant changes in the performance when we changed the pixel size of images. The performance increases when we decrease the image size from 128 × 128 to 64 × 64 and then to 32 × 32.

Image classification using machine learning requires feature extraction for better performance whereas image classification using deep learning extracts the feature by itself. We have applied Gabor filters and Sobel kernel for feature extraction and edge detection. The performance of the ML classifiers can be further improved by applying more such techniques along with dimension reduction, but it still cannot be compared with the performance of deep learning methods.

Thus, this project shows the advantages as well as limitations of traditional ML techniques in image classification and proves the need for deep learning techniques to solve such problems to achieve accuracy greater than 90%.

***

References

- https://www.youtube.com/channel/UC34rW-HtPJulxr5wp2Xa04w
- https://towardsdatascience.com/
- https://www.javatpoint.com/machine-learning
- Comparative analysis of image classification algorithms based on traditional machine learning and deep learning : https://www.sciencedirect.com/science/article/pii/S0167865520302981
- Image Classification using SVM and CNN : https://ieeexplore.ieee.org/abstract/document/9132851/

***