# "Image Classification"

by Gurusamy, Brinda & Baldwin, John, University of California, Berkeley

Dec 2019

## 1. Abstract

Image classification is a machine learning problem that uses computer vision to extract features from images and uses a combination of extracted features to classify and categorize images. In this paper, we are using a dataset of 1501 labelled images belonging to 20 different categories. Features were extracted from images in the dataset and given as input to classification algorithms like logistic regression, k-nearest neighbors, random forests, decision trees, and support vector machines. The dataset was split into training and testing datasets, and 5-fold cross validation and regularization were applied to the above mentioned classification algorithms and the results showed that support vector machine algorithm performed the best giving a test accuracy of 46%.

## 2. Introduction:

Compared to how easy it is for humans to identify and classify images, computers and machine learning algorithms need features that distinctly separate one image from another to classify them. This paper focuses on the complete lifecycle of an image classification problem.

## 3. Description of Data:

For this image classification problem, we used a training image dataset consisting of 1501 images and validation image dataset consisting of 716 images, belonging to 20 different categories. Categories from the dataset include ‘Airplanes’, ‘Comet’, ‘Zebra’,’Penguin’ etc. The dataset has different number of images for different categories as shown in the graph below:

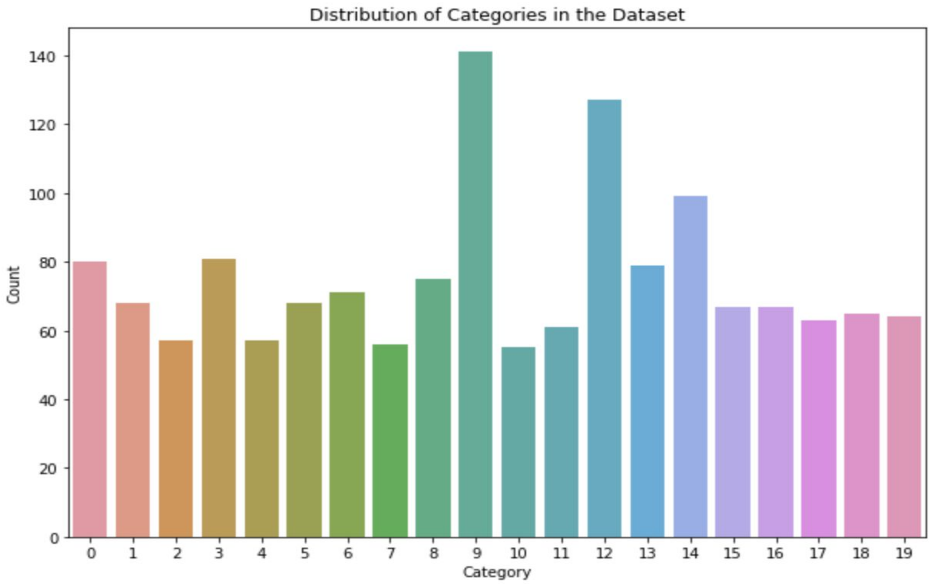

Fig 1. Graph showing distribution of categories in the dataset

The training data images were converted into a 3-dimensional numpy array and the categories were encoded with integer ranging from 0-19 before reading into a dataframe.

## 4. Description of Methods:

### 4.1 Data Cleaning and Transformation

The training dataset had no missing values or any values that were out of range. Hence, data cleaning was not required. However, we found that some images were on greyscale and required data transformation. The grayscale images were identified and **transformed into RGB images** to maintain uniform dimensions and overcome problems during feature extraction.

### 4.2 Feature Engineering

The images in our dataframe were expressed as 3-d numpy arrays and this format was not a suitable input for classification algorithms. Some reasons why a 3-d numpy array was not suitable are:

1) The number of features/ dimensions will be high
2) As each image varies in size, the number of dimensions will not be constant across images.

Due to the above mentioned reasons, we had to extract suitable features from images that can be used as input to the classification algorithms. These features could either be **scalar values or matrix based vectors** of equal sizes or varied sizes.

### 4.3 Scalar Features

Scalar features are the ones that represent an image by a single value. The list of scalar features that were extracted from training images can be found in the table below:

Table 1. Description of scalar features considered for the model

| Feature name | Feature Description | Included or Excluded |
|---|---|---|
| Image size | Number of pixels in an image | Included |
| Aspect ratio | The ratio of width to height of an image | Included |
| Mean red channel intensity | The mean of red channel intensity of all pixels in the image | Included |
| Mean green channel intensity | The mean of green channel intensity of all pixels in the image | Included |

| | | |
|---|---|---|
| Mean blue channel intensity | The mean of blue channel intensity of all pixels in the image | Included |
| Standard deviation of green channel intensity | The standard deviation of green channel intensity of all pixels in the image | Included |
| Standard deviation of red channel intensity | The standard deviation of red channel intensity of all pixels in the image | Included |
| Standard deviation of blue channel intensity | The standard deviation of blue channel intensity of all pixels in the image | Included |
| Standard deviation of hue values | The standard deviation of  hue values of all pixels in the image | Included |
| Standard deviation of saturation values | The standard deviation of saturation values of all pixels in the image | Included |
| Standard deviation of brightness values | The standard deviation of brightness values of all pixels in the image | Included |
| Number of blobs | Blobs in images are a group of connected pixels and only large groups are considered as blobs and small ones are disregarded as noise. | Excluded |
| Contour extent | To calculate this feature, we identify the second largest contour in the image and draw a bounding rectangle around it and calculate the percentage of contour area vs the bounding rectangle area. | Excluded |
| Mean Grayscale intensity | The mean of grayscale values of all pixels. | Included |

### 4.3.1 Exploratory Data Analysis of Scalar Features

In the exploratory data analysis, we'll analyse how the different scalar features vary across different categories. By using a similar approach shown in the three graphs below, all scalar features were explored to see how they vary across different categories. The features that vary more across categories are chosen to be used in the model.

### 4.3.1.1 Image Size vs Category

The image size varied across different categories and can be considered a good feature to include as an input to the model.

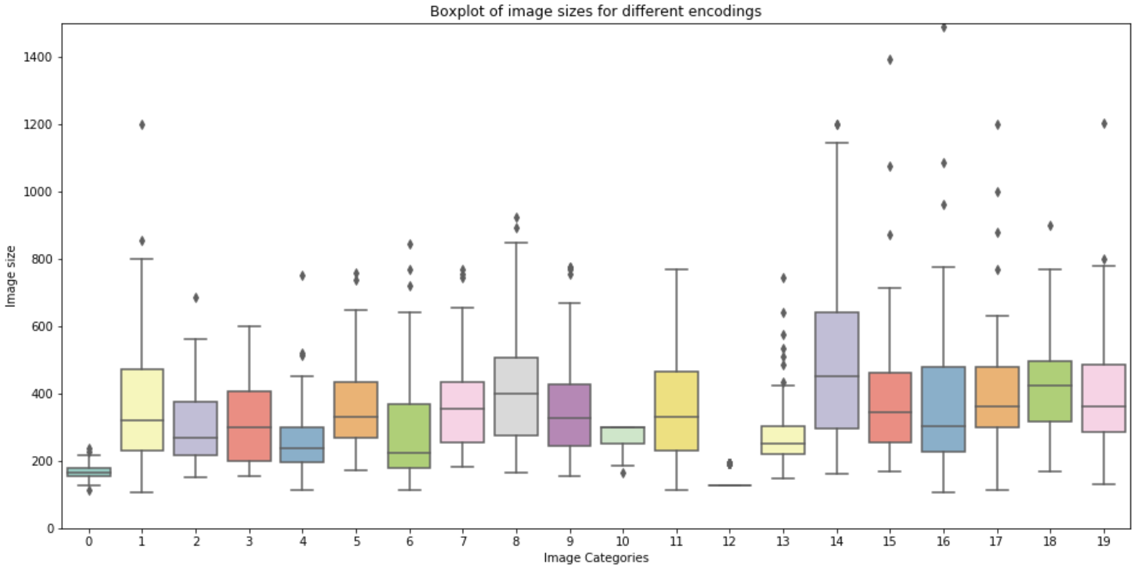

Fig 2. Graph showing the variation of image sizes across categories

### 4.3.1.2 Mean Grayscale Intensity of images vs Category

The grayscale intensities also seem to vary across different categories and can be used as an input to our machine learning model.

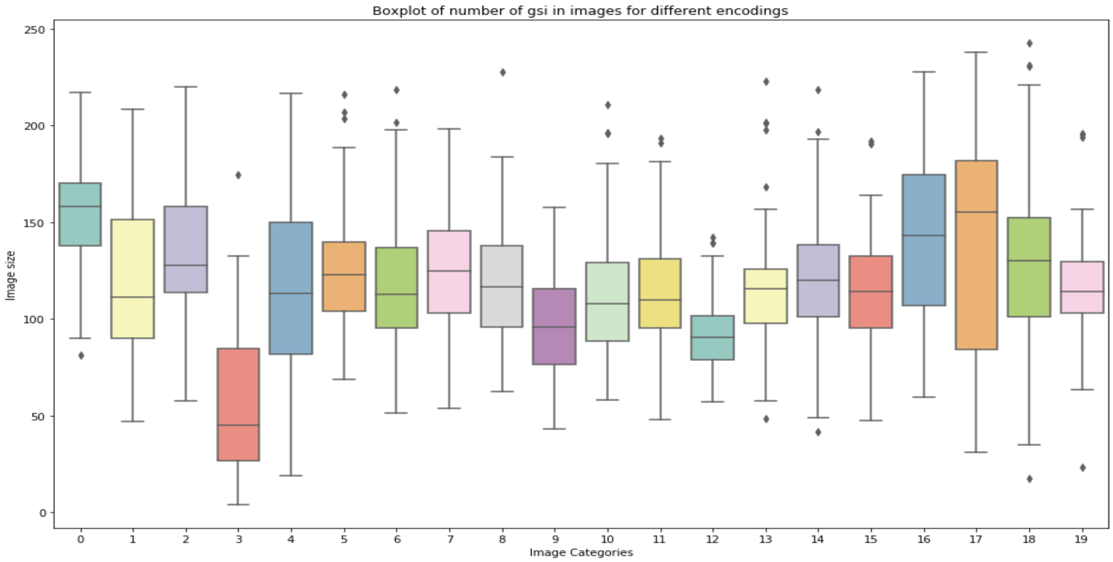

Fig 3. Graph showing the variation of grayscale intensity mean across categories

#### 4.3.1.3 Mean green channel intensity vs Category

The mean green channel intensity seem to vary across some categories if not all and hence can be used as an input to the model.

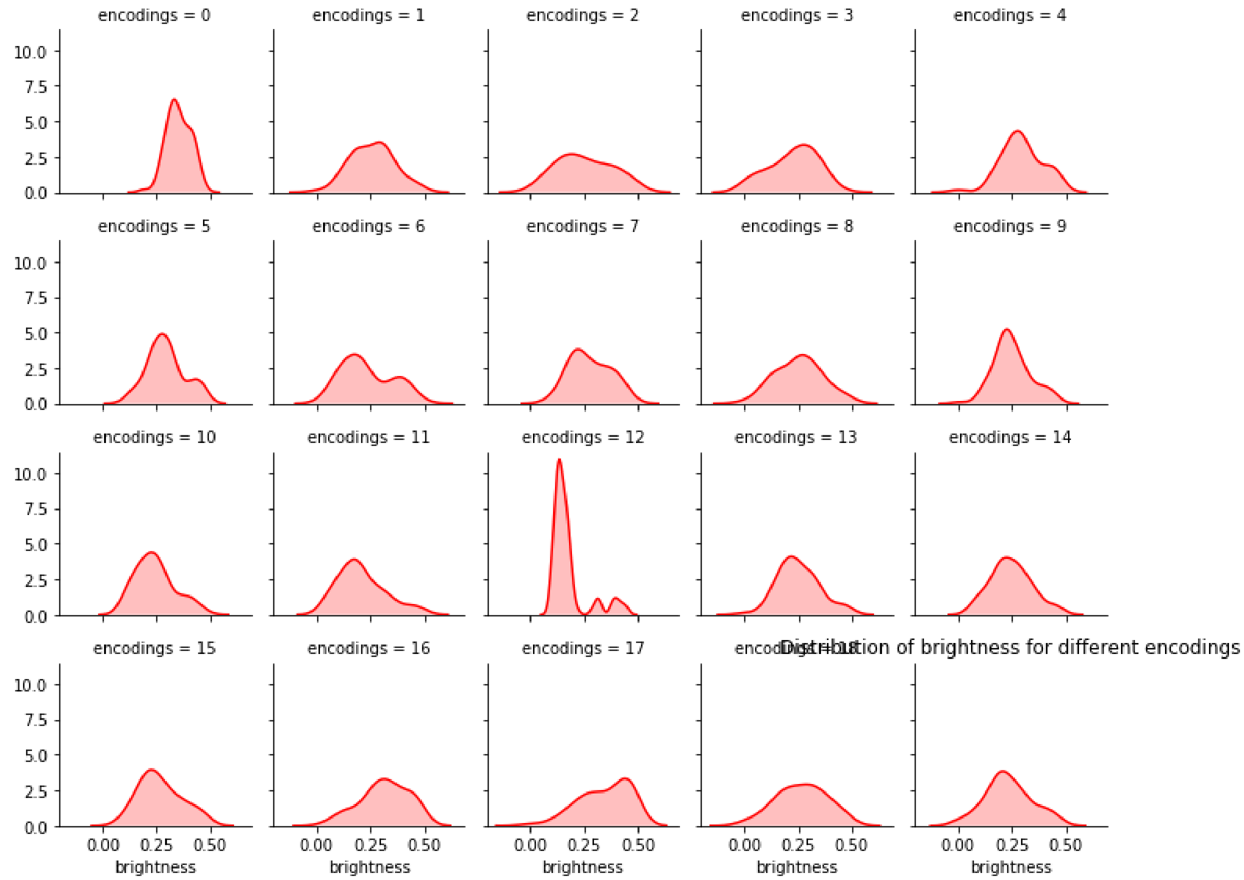

Fig 4. Graph showing the variation of mean green channel intensity across categories

### 4.4 Matrix-based/Vector Features:

The matrix or vector features provide multidimensional values or vector values**.** The following table shows the description of the vector features and their inclusion status. These features were either flattened or reduced in dimensions to be used as input.

Table 2. List of matrix/vector features considered for the model

| Features | Feature Description | Included or Excluded |
|---|---|---|
| Color Histogram | A histogram showing the distribution of colors in an image | Included |
| Haralick textures | Quantifies the texture of the image | Included |

| | | |
|---|---|---|
| Hog( Histogram of Gradients) | Distribution of gradient directions in the image.<br>The hog feature vector when flattened had multiple dimensions and hence dimensionality reduction(PCA) was applied to get the most useful features. | Included |
| Canny Edge Detection | Gives the multiple edges in the images. Since this feature also gave huge multi-dimensional vector, dimensionality reduction was applied to get the most useful features. | Included |
| Hu Moments | The hu moments are weighted averages of the pixel intensities of the image. | Excluded |

### 4.5 Classification Algorithms:

After the features were constructed and added to the dataframe, we split the entire dataset into training and testing data. The training data will be evaluated based on five classification algorithms: Logistic regression, Random forests, decision trees, k-nearest neighbors, and support vector machines. K-Fold cross validation was applied to the training data and for each classifier the hyperparameter was tuned to avoid overfitting and an optimal model parameters were acquired. The test data was ran on the optimal models each classifier and here is the summary of accuracy for different models.

Table 3. Accuracy & Hyperparameter values of different classifiers

| | Model | Test Accuracy | Tuned Hyperparameter |
|---|---|---|---|
| 1 | Logistic Regression | 0.44680851063829785 | C = 12 |
| 2 | Random Forests | 0.425531914893617 | N_estimators =300 |
| 3 | Decision trees | 0.2978723404255319 | Max_depth = 13 |
| 4 | K-nearest neighbors | 0.3404255319148936 | - |
| 5 | Support Vector Machine(Highest Accuracy) | 0.4601063829787234 | C = 0.6 |

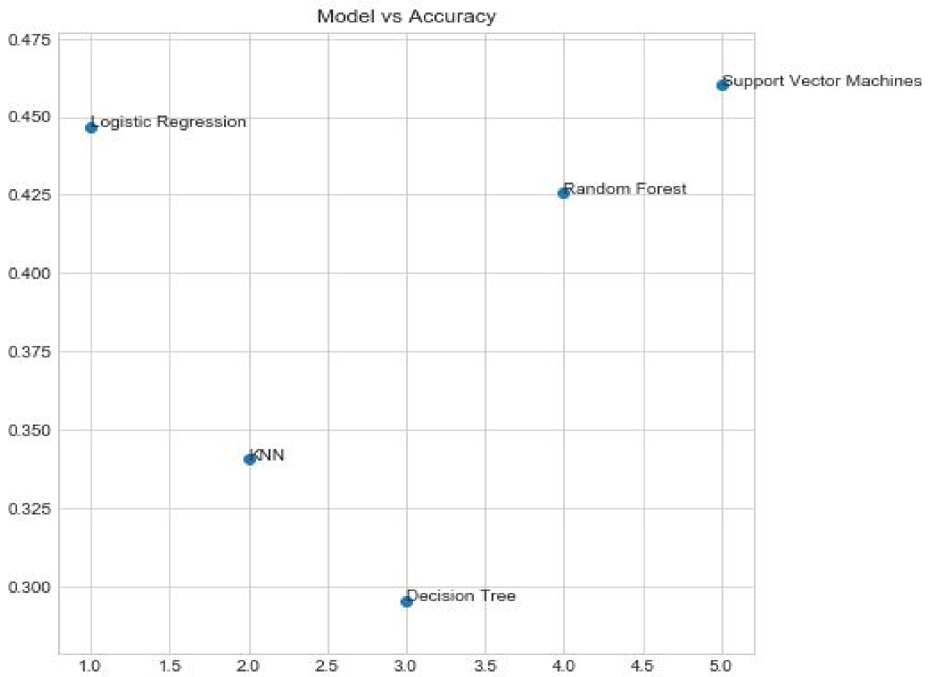

Fig 5. Graph showing accuracy of different classifiers

## 5. Discussion:

The most interesting discovery in the analysis was how some simple scalar features like image size and red, green, blue channel intensity, grayscale intensity means can provide significant accuracy and these features also varied well across multiple categories. Some features  like blobs, contour extent, hu moments seemed very promising at first, and varied well across different categories but added little improvement to the model accuracy. We believe this is because some features closely resembles another another feature. For example: “Hu moments” calculates based on the weighted average image channel densities which is very similar to mean grayscale image pixel intensity or mean RGB channel intensity mean. Hencing including the “Hu moments” feature doesn’t provide any increase in improvement when scalar features like RGB channel intensity mean are used.

## 6. Summary of Results:

Among the machine learning models, random forests, logistic regression, and support vector machines were the three models that performed better than the others like the Decision trees and k-nearest neighbors. After performing 5-fold validation and regularisation, “Support Vector Machines(SVM)” gave the highest accuracy of 46% on the test dataset. The currently used model classified images very well based on color pattern recognition and matching. The future work will concentrate on making improvements on how well the model can pick up shapes and not reply only on color patterns to make predictions. We also plan to use convolution neural network on this dataset and evaluate its performance in the future.