# A Computational Study on Image Classification

Ashish Rawat B01504783

May 11, 2018

**Abstract**

The field of computer vision is an interdisciplinary field concerned with scientific methods and technology of building artificial systems that can interpret meaningful information from the physical world from images or visual sensors. Object recognition is an important field of computer vision which deals with finding and identifying objects in images by using image classification. In this paper we discuss a brief history of image classification and evaluate some classification methods on a particular image data set. In the end we present our evaluation on the current state of the art image classification model based on Deep Neural Networks and Convolution Neural Networks.

## 1. Introduction

The objective of a classification model is to assign a given sample of unknown class or category, to its correct class; for example, an object like a 'telephone' may represents categories of 'device', 'electronics', 'technology' or 'communication'. According to LA times, an average US household contains close to 300,000 items [2], which raises the question of how many visual categories/classes are there that are required to be modeled in order to achieve successful classification. What makes the classification task even more challenging is the variance of objects within a class label; for example a class representing 'chair' will have to deal with representing chairs of all shapes, sizes and designs. Besides this, it is often the case that two images might contain

the same objects under different lighting conditions, making them look substantially different from one another. Other variances include positioning, rotation and mirroring of the objects in an image, and sometimes another object might occlude a part of the object of our interest. Primitive approaches for recognizing objects in images often suffered from these ailments, however, new technologies emerged with time handling some hurdles such as rotation and invariance.

The earliest object recognition techniques were inspired by recognition using parts, based on techniques such as generalized cylinders and geons [2]. Geons are simple forms such as cylinders, circles, rectangles, cones etc. which form simple parts of an object in an image. In 1987, Irving Biederman presented his recognition by parts theory where he suggested that there are around 36 geons, and various combinations of these geons can represent all the objects that we see in our day to day life [3]; for example a bus can be represented in a two dimensional world 'tiny world' or 'blocks world' by a rectangle placed on top of two circles. The main strength of this approach was that an object could be recognized regardless of viewing angle (Viewpoint Invariant) when represented as a combination geons, however the theory provided no methods of deriving geons from an image[4], and experiments have shown that it was difficult to extract such parts from an image[1].

New technologies based on edge detection were developed, and edges in an image became the most distinct and useful features for object recognition. Various approaches were developed that attempted at extracting meaningful information from images. Some approaches combined both low level features such as edges and SIFT, which detects local features in an image [1] as well as global features like Histogram Oriented Gradients (HOG). A significant improvement was the use of the bag of features model which was analogous to the bag of words model used for text classification.

We now present our evaluation of some classification methods for image classification. We begin by evaluating the performance of classification by matching local feature between images, moving on a generative classifier such as Naïve Bayes and Linear Discriminant Analysis. After this we will use some common Machine Learning models such as Logistic Regression, Decision Trees and Ensemble Learning for

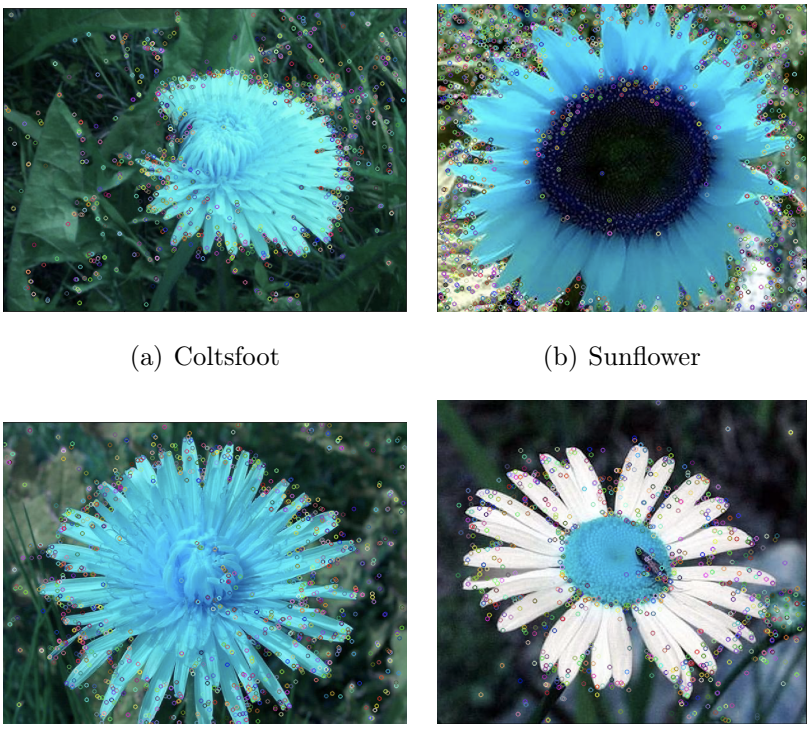
image classification, finally presenting the current techniques based on neural networks. For the purpose of this study we shall be using a dataset called the 'Flower 17' dataset, which contains 17 different classes of flowers with each class containing 80 example images.

## 2. Image classification using local features.

In this approach we use brute force feature matching between a sample image and the given training data. We first begin by processing all the images in the training data one by one, extracting local features of each image and creating a feature vector, creating a database of feature descriptors (vectors) for each image in a class and storing those features grouped by the particular class name. For extracting features, we use a feature detector algorithm like SIFT, which outputs locations of significant areas in the input image. In our implementation we use KAZE feature description, the key points generated by the descriptor can be visualized in Figure 1.

As for our classification process, for an input image we first compute the local features for the image representing it as a vector of features. We then iterate over all the training data, calculating a similarity score between the input image feature vector and the training image feature vector. Our similarity score is based on the cosine distance between the two vectors. We find out the classes of the top 5 training vectors of highest similarity score, and the majority class among the 5 classes is our predicted class value.

The results of classification using local feature matching were quite poor, the primary reason for this poor performance was the similarity of images in our dataset. Flower species, although differentiable by human eye, could not be handled optimally by an algorithm that used local features of images. This shortcoming was aggravated by the fact that our approach did not use any kind of foreground extraction or image segmentation, and as a result we can visualize from fig that even background features are also being recorded and used in matching.



(a) Coltsfoot (b) Sunflower

(c) Dandelion (d) Daisy

Figure 1: Visualization of KAZE feature descriptors extracted from Images

## 3. Generative Classifier- Naive Bayes

In the previous section, we used local features of an image for classification, however local features points are insufficient for representing an image as a whole. Also, local feature descriptors that we used did not take into account important characteristics such as color while quantifying an image. As opposed to local features, global features of an image quantify image as a whole. Color, edges and texture are important image descriptors that allow for successful classification of images and can collectively act as global features of an image. In order to use multiple features of an image for representation, we can use a technique call 'Bag of Visual Words'(BoVW). Inspired from bag of words(BoW) technique in the field of text classification, where BoW is a vector representing counts of words in a document, BoVW is a vector of counts of image features [5]. For our classification task, we extract color, texture and shape features of an image and group them together to form a bag of visual words,

in other words a vector representing global features of an image. Now we present two generative models for image classification namely the Naïve Bayes Classifier and the Linear Discriminant Classifier.

Naïve Bayes classifier is a probabilistic classifier based on Bayes Rule, that assigns class labels to input instances. Let our feature vector be represented as $\mathbf{x} = \{x_1, x_2, .., x_n\}$, and our model consists of $\mathbf{k}$ classes represented by $c_1, c_2, .., c_k$, then the model assigns probabilities to each class $P(c_k|x)$. Now using Bayes rule we can represent $P(c_k|x)$ as $P(c_k)P(\mathbf{x}|c_k)$. Naive Bayes assumes that each feature $x_i$ is independent of other features in the vector which reduces to $P(c_k)\Pi_i P(x_i|c_k)$. So now our prediction rule becomes as follows-

$$\hat{c} = argmax\ _C\ P(c|\mathbf{x}) = argmax\ _C\ P(c) \prod iP(x_i|c)$$

Representing our image as a BoW/vector allows us to use Naïve Bayes classifier for predicting the class of the image based upon the prediction rule presented above [6].The dataset was split into training set (70%) and test set (30%), and the classifier was able to obtain an accuracy of only 37% on the test data. Accuracy is defined as the precentage of samples labeled correctly by the classifier. We also derive the metrics of Precision and Recall from the models confusion matrix. Table 2. represents the confusion matrix for the Naive Bayes classifier, where the major diagonal represents the number of correctly classified data. Table 1. shows the Precision, Recall and Accuracy of the model.

Table 1: Precision, Recall and Accuracy for Naive Bayes Classifier

| **Model** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|
| Naive Bayes | 0.39 | 0.33 | 0.37 |

Being a generative model, Naïve Bayes classifier makes explicit claims about how the training data is being generated, which might work against us in cases like image classification where there is not actual underlying process that governs the generation of the training data. Another drawback is the strong assumption that

feature points in a vector are independent of each other, which leads to the classifier ignoring the spatial relationships among features in a vector, which is an important metric for an image classification task.

Table 2: Confusion Matrix for Naive Bayes Classifier. The first row represent the actual class of the data and the first column represent the predicted class of the data.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 10 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 1 | 4 | 0 | 3 | 0 | 0 | 1 | 0 | 0 |
| **2** | 0 | 8 | 0 | 2 | 0 | 4 | 1 | 8 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 0 |
| **3** | 0 | 0 | 9 | 1 | 3 | 2 | 0 | 2 | 0 | 0 | 0 | 3 | 0 | 1 | 0 | 0 | 0 |
| **4** | 0 | 1 | 0 | 6 | 1 | 3 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 5 | 1 |
| **5** | 1 | 0 | 0 | 1 | 9 | 0 | 0 | 0 | 0 | 4 | 0 | 4 | 0 | 0 | 0 | 2 | 1 |
| **6** | 0 | 3 | 7 | 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 5 | 0 |
| **7** | 0 | 0 | 0 | 2 | 1 | 0 | 11 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 6 |
| **8** | 0 | 3 | 1 | 3 | 0 | 2 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 |
| **9** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 15 | 4 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| **10** | 2 | 0 | 1 | 0 | 4 | 0 | 1 | 1 | 3 | 6 | 0 | 5 | 0 | 1 | 3 | 1 | 1 |
| **11** | 1 | 3 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 6 | 0 | 5 | 0 | 3 | 2 | 5 |
| **12** | 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 5 | 0 | 8 | 0 | 0 | 0 | 0 | 0 |
| **13** | 1 | 0 | 1 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 5 | 0 | 0 | 6 | 6 |
| **14** | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 8 | 4 | 3 | 0 |
| **15** | 0 | 0 | 0 | 2 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 14 | 2 | 0 |
| **16** | 0 | 3 | 2 | 7 | 2 | 9 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 4 | 0 |
| **17** | 0 | 0 | 0 | 0 | 0 | 1 | 5 | 0 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 4 | 6 |

# 4. Classification using traditional Machine Learning Methods

Unlike the generative classifiers, deterministic classifiers are only concerned with separating the given data into their respective classes and thus don't attempt to model the underlying data generative process. The assumption that features are independent of each other no longer holds in case of deterministic classifiers and in fact these models derive their strength from the relative relationships between features. We have evaluated our dataset with two popular classifiers namely Decision Trees, Logistic Regression, as well as an ensemble classifier based on Random Forests.

A decision tree classifier constructs a decision tree from the training data, where each node in the tree represents a particular feature/input variable and the edges from a node correspond to the various values that the features can take. The leaves of the decision tree are the representation of the target classes.

Logistic Regression classifier separates the given data into different regions where each region represents a particular class. Logistic regression the predicts the probabilities of an input sample belonging to one of these classes. The class with the highest probability is the class closest to the input sample.

Table 3: Precision, Recall and Accuracy for Decision Tree, Logistic Regression and Random Forest models.

| **Model** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|
| Decision Trees | 0.38 | 0.38 | 0.44 |
| Logistic Regression | 0.35 | 0.36 | 0.50 |
| Random Forests | 0.55 | 0.54 | 0.65 |

The third model is an Ensemble learning model based on Decision Trees. Ensemble learning methods use multiple learning models to boost predictive performance. We use a Random Forests as our ensemble learning model, which is a collection of many Decision Trees. The Random Forest algorithm builds multiple decision trees

at the time of training, and during prediction the it outputs the mode of classes predicted by the collection of trees. The Random Forest model that we implemented used 100 Decision Trees for classification.

Table 3. lists out the precision, recall and accuracy of the three models that we have studied. It can be observed that only Random Forests perform well among these three models, however the accuracy of 65% is still not satisfactory. Table 4. depicts the confusion matrix for the Random Forest model.

Traditional machine learning methods suffer when the dimensionality of the data is very large, meaning that the sample contains a large number of features. Such is our case, where each sample contains a large number of features describing the color, shape and texture of features.

Table 4: Confusion Matrix for Random Forest model.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 18 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 3 | 0 |
| **3** | 0 | 0 | 15 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| **4** | 0 | 4 | 0 | 7 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 6 | 0 |
| **5** | 0 | 0 | 1 | 0 | 12 | 0 | 3 | 0 | 0 | 1 | 1 | 2 | 1 | 0 | 0 | 0 | 1 |
| **6** | 0 | 5 | 1 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 1 | 7 | 0 |
| **7** | 0 | 1 | 0 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| **8** | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 5 | 0 |
| **9** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| **10** | 9 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 10 | 2 | 1 | 1 | 0 | 0 | 0 | 3 |
| **11** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 20 | 0 | 7 | 0 | 0 | 0 | 1 |
| **12** | 3 | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 1 | 1 | 0 | 8 | 0 | 0 | 0 | 0 | 0 |
| **13** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 8 | 0 | 15 | 0 | 0 | 0 | 1 |
| **14** | 0 | 0 | 3 | 0 | 0 | 1 | 0 | 4 | 0 | 1 | 0 | 0 | 0 | 13 | 1 | 2 | 0 |
| **15** | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 14 | 1 | 0 |
| **16** | 0 | 6 | 4 | 4 | 1 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 |
| **17** | 0 | 0 | 0 | 0 | 1 | 0 | 4 | 0 | 0 | 1 | 3 | 0 | 4 | 0 | 0 | 1 | 6 |

# 5. Image classification and Convolutional Neural Networks

We have seen how traditional classification techniques are not capable of handling the task of image classification at all. Even with all the feature engineering and extraction, the classifiers fail to successfully separate data. In this section we will evaluate the neural network approach to image classification. We will look at Convolutional Neural Networks and discuss how they classify images and then present the result of classifying the flower dataset using a Convolutional Neural Network (CNN). Unlike previous approaches that we have used, there is no requirement for extraction of features before running the classification algorithm, so input to our neural network is always raw image.

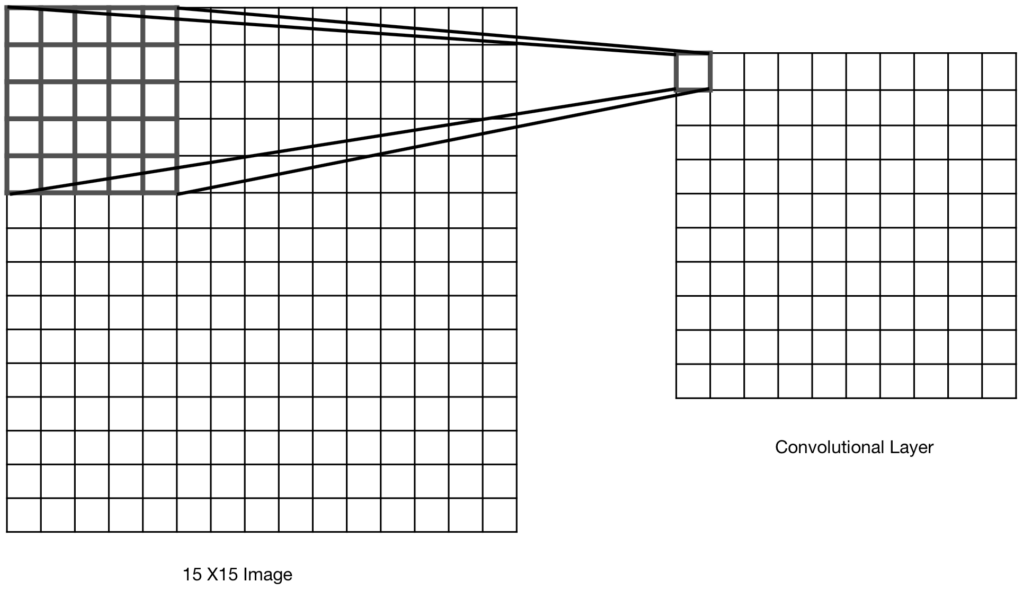



Figure 2: A depiction of the process of convolution in a Convolutional Neural Network

Convolutional Neural Networks work by passing the image through a series of convolutional and pooling layers to get an intermediate output. This intermediate

output can be thought of as a feature descriptor of the image, which is then passed on to the final classification layer. The process of convolution involves scanning parts of an image one at a time by use of a filter, sometimes called a kernel. The part of the image that the filter is currently scanning is called the receptive field of the filter. The filter reads the pixel values from its receptive field and multiplies it with the values in the filter and computes one numerical value, which can be thought of as a feature of the image. The filter then slides over the entire image pixel by pixel and computes a numerical value at each step. So if our image is 15 by 15 pixels, and the filter is 5 by 5 pixels then the output of convolution would be an array of 10 by 10 numeric values called an activation map. Depending on the number of filters used, we can have multiple activation maps after the first convolution step and a CNN goes through multiple cycles of convolutions. Figure 2. depicts the processing in the first layer of a CNN.

At an abstract level, the filter can be thought of as a feature detector, so we might have multiple filters for detecting different features such as colors, shapes and edges. The filters at the first few layers could be thought of as local feature extractors. The filters at higher levels compute features from the activation maps produced by the lower layers, and can be thought of as global feature extractors. The last layer of the network is basically the classification layer, which assigns probabilities to an vector representing the classes in our model.

For the purpose of this study we have used the MobileNet, a lightweight CNN architecture developed by Google for image classification [9]. We used transfer learning, a technique of retraining a piece of a model that has already been trained on a related task and reusing it in a new model[10], since training the model from scratch would be a very time consuming and computationally expensive task. The results were far better in comparison to the earlier models; this model achieved an accuracy of 93% on test data. The precision, recall and Accuracy are listed in Table 5 . The confusion matrix for the CNN model in Table 6. shows that almost all the data was classified according to correct class labels.

Table 5: Precision, Recall and Accuracy using MobileNet.

| **Model** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|
| CNN | 0.94 | 0.94 | 0.92 |

Table 6: Confusion Matrix for prediction of flowers using MobileNet CNN

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| **3** | 0 | 1 | 20 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 1 | 0 | 0 | 0 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| **6** | 0 | 0 | 0 | 2 | 0 | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **9** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 23 | 0 | 0 | 0 | 0 | 0 | 0 |
| **12** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 23 | 0 | 0 | 0 | 0 | 0 |
| **13** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 21 | 0 | 0 | 1 | 0 |
| **14** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24 | 0 | 0 | 0 |
| **15** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 20 | 1 | 0 |
| **16** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 23 | 0 |
| **17** | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 21 |

# 6. Conclusion

CNNs are able to handle very large feature vectors, for instance InceptionV3 can handle vectors of about 100000 features, MobileNet It is evident that connectionist models have much greater performance than the traditional classification models, and are successful not only in differentiating among just seventeen classes of flowers

but among hundreds of classes. Some architectures that use a hybrid of traditional methods for classification and neural networks for feature detection, are also able to perform much better than methods that do no use neural networks for feature extractions. Experiments have shown that using Logistic Regression after extracting features using CNNs gives results comparable to what we observed using the fully connected MobileNet CNN[10]. A downside of using CNN is that they require high computational power and large amount s of data for training; their performance is directly related to the amount of data available.

# References


[1] Alexander Andreopoulos *50 Years of object recognition:Directions forward*

[2] LA TIMES, By Mary MacVean

[3] Irving Biederman *Recognition-by-Components: A Theory of Human Image Understanding*

[4] Recognition-by-components theory
`https://en.wikipedia.org/wiki/Recognition-by-components_theory`

[5] Bag-of-words model in computer vision
`https://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision`

[6] Naive Bayes classifier
`https://en.wikipedia.org/wiki/Naive_Bayes_classifier`

[7] Sachin Joglekar: Logistic Regression
`https://codesachin.wordpress.com`

[8] Understanding Convolutional Neural Netowrks
`https://adeshpande3.github.io`

[9] Andrew G. Howard *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*

[10] How to Retrain an Image Classifier for New Categories
https://www.tensorflow.org/tutorials/image_retraining

[11] Flower Recognition using Deep Learning https://gogul09.github.io