VRIJE
UNIVERSITEIT
BRUSSEL

PROJECT REPORT

Machine Learning

Anisha Sachdeva
anisha.sachdeva@vub.be
0571873

January 17, 2021

Prof. Ann Nowe

1. Introduction

Image classification is a technique used in machine learning to classify an image belonging to a particular class when a number of predefined classes are given. The goal of our project is to create a classifier which would most accurately classify an animal to a particular class, while 12 distinct classes of animals are given. We will be applying various image classification techniques along with different pre-processing procedure in order to create the best classifier. In the following section, we will discuss about the methodology used followed by result and discussions.

2. Methodology

The following methodology was used to build a machine learning pipeline wherein we tested different statistical machine learning classifiers with varying pre-processing techniques and feature descriptors in order to test the performance of each algorithm and finally selected the best one.

2.1 Data Analysis

We first analysed our data by loading the images from the training dataset. We noted that 12 classes of animals were specified with labelled folders (each folder for each species), helping us to infer that the problem given to us is a supervised classification problem. The 12 classes were as follows:

1. Chicken	7. Jaguar
2. Elephant	8. Lion
3. Fox	9. Owl
4. German Shepherd	10. Parrot
5. Golden Retrieve	11. Swan
6. Horse	12. Tiger

In another folder, we were given unlabelled test images (i.e. all kind of animals in the same folder) for which we had to test the accuracy of our best fit model/ image classifier. In total we had 4042 labelled images for training and 4035 unlabelled images for testing.

Next, we plotted first four images of each class from the training dataset to get a look and feel of our dataset. We noted that the data provided to us is diverse and has noise with it. We inferred that as next steps, we will be pre-processing the images in order to reduce the noise present in the training data, followed by different feature extraction techniques.

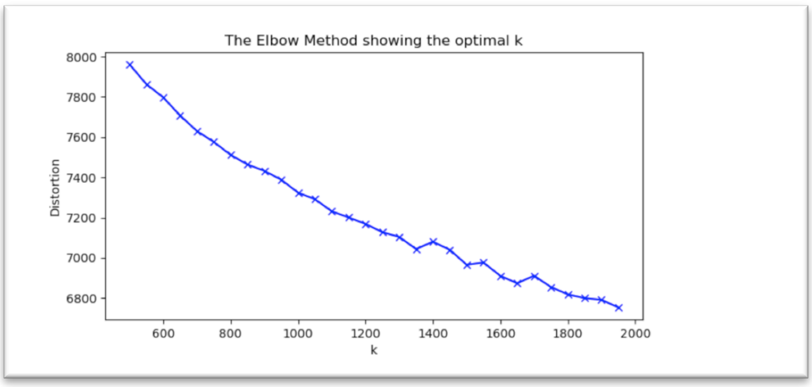
To kick start our project, we selected the following five most popular supervised classification algorithms, with which we would be playing around in terms of testing different hyper-parameters with them or testing the algorithms with different feature descriptors and pre-processing techniques.

1. **Naïve Bayes:** This algorithm is based on the Bayes theorem with the assumption that every pair of features which is classified behaves independent of each other. This algorithm does not require a lot of training data but at the same time it is known as being a bad estimator.
2. **K-Nearest Neighbours:** As the name suggests, this algorithm classifies on basis of similarity and majority votes with the nearest neighbours. This is a lazy and instance-based algorithm whose accuracy might increase, when the number of neighbours i.e., value of k increases.
3. **Decision Tree:** This algorithm predicts the class of the test images by learning some simple decision rules deduced from the training data.
4. **Random Forest:** In random forest, forest means a group of trees and random refers to selecting random data. This type of classifier selects random data and then forms decision trees of them with getting prediction values from each tree. The best solution is based on voting of prediction values by each tree in the forest.
5. **Support Vector Machine:** SVM uses some extraordinary complex data transformations based on which it then finds an optimal boundary between the possible outcomes. In a decision hyper plane, SVM separates the objects belonging to different classes and it tries to maximize the margin between them with support vector's help. SVM used the five-fold cross validation technique which helps to classify the data in the best possible manner.

Further, as provided to us, we had seven types of different feature descriptors namely:

1. **Boost:** This is an image descriptor which learns with boosting
2. **Freak:** Biologically motivated descriptor
3. **Daisy:** This descriptor depends on histogram of gradients and is very fast and powerful to compute
4. **Lucid**
5. **ORB:** ORB stands for Oriented fast and Rotated Brief feature detector which is rotation invariant and is robust.
6. **SIFT:** They are in general considered as the best feature descriptors as very fast and robust in nature.
7. **VGG**

We know once the features are extracted, they are clustered together bases on similarities they share. In our experiment, we used K-means clustering wherein we specify the number of clusters (codebooks) in such a way that the error (or variation) within the cluster is minimum. Initially in the starter's kit we were provided with the codebook size as 500. To verify if 500 was an optimal value for k, or if we could have a better k-value, we used the elbow method. We specified the range of the codebook size from 500-2000. With the graph obtained from the elbow method, the point where there is a sharp elbow that point is considered as the optimal value for k.



We did not get a very clear elbow, however we could observe from the graph that 1700 would serve as a good value for k. Hence, throughout our experiment, we considered the codebook size as 1700.

Now we had the rough framework consisting of various classifiers, feature descriptors and an optimal value for clustering the features. We needed to check which combination of features and classifiers, along with some pre-processing would be best suited for our experiment. We prepared a project plan which was divided into three levels.

1. **Level 1:** In level 1, we decided to shortlist the two best supervised classifiers out the five selected as mentioned above. Here, we split the training dataset into training and validation datasets. In order to have a fair comparison of all the classifiers, we used the raw images as provided to us and extracted the features followed by clustering of the features together. We then modelled each of our classifier and tested their accuracies. Based on the best accuracy scores we selected two best classifiers with the best two feature descriptors.
2. **Level 2:** In level 2, we used some pre-processing techniques to validate how our selected classifiers perform when the images are resized or some other pre-processing has been done.
3. **Level 3:** In level 3, we selected our best performing classifier under different circumstances and here we tried to improve its accuracy with some other pre-processing techniques.

In general, all our levels in the experiment contain the following steps:

1. **Image pre-processing:** This step was performed to refine the features of an image by suppressing the unwanted noise and enhancing some important features in order to better train our models and improving their performance.
2. **Training data split:** In this step, we split the train data into train and validation datasets. Our model is trained on the training dataset, whereas it's performance or accuracy is measured on the basis of how well the model could classify the images for the validation dataset.
3. **Feature extraction:** In our experiment, we are using bag of visual words (BoW) model wherein a train image is presented as a set of features and the features contain descriptors which are unique to an image. The similar kind of features are then clustered together which would be unique for a particular class helping the model to train and differentiate between two classes.
4. **Training the model:** Using different algorithms like SVM, Decision Tree, the model was trained meaning that the model learned about the features from the labelled data.
5. **Checking the accuracy of the model:** Once the model is trained, it can be fed with the validation data which is completely new and never seen by the model. The model then tries to classify the images and the accuracy is measured as the ratio of correctly predicted outputs to the total number of outputs.

2.2 Experiment: Level 1

1. **Image pre-processing:** For the initial step, we did not perform any pre-processing instead used the raw images. This was done because of the fact that if any pre-processing technique such as resizing, gaussian blur, median filtering would have been used, then it might would have been unfair for some of the classification algorithms as different algorithms behave differently to such techniques (and that too for given different features). All pre-processing techniques do not help in providing only the best results for each of the algorithm. We decided that once, we would get an overview of general performance of each algorithm with all the different types of features, we will then select the best two algorithms and for them we will try different pre-processing techniques.
2. **Training data split:** We split out training data in the ratio of 80-20. 80% of the training dataset i.e., 3233 images were used to train the model and the 20% of the remaining i.e., 809 images were left to validate how well our model is trained and can classify correctly. The validating dataset behaves as an unbiased dataset as the data fed to the model while testing its accuracy is all new and never seen before. *(We even tried the split ratio as 75-25 but we noted that there was not much difference between the accuracy score provided by the validating dataset)*
3. **Feature extraction:** For the initial step, we did not select any one random feature, because there is no such thing as the 'best feature descriptor'. The quality of the feature descriptor depends on the image and we wanted to check which feature works best with which classifier hence, we extracted all the seven features for all the five kind of classifiers. K-means clustering was used with codebook size as 1700.
4. **Training the model:** We trained each of the five models with all the seven features extracted. The default hyper-parameter values were considered for all the models.
5. **Checking the accuracy of the model:** Now for all the five types of classification algorithms with seven types of feature descriptor, we evaluated the performance of each model by first training the model on the training dataset (i.e. 80% of the original training data) and calculating the accuracy score on the validating dataset (i.e. 20% of the original training data which was not involved while training the model). We computed and compared the performance of each algorithm on the basis of the three below mentioned parameters:

    i. Accuracy score: This score is based on the ratio of correctly predicted observed images to the total number of images.
    ii. Confusion matrix: This is a type of 2-D matrix which describes the performance of a classifier on the validation dataset for which the true values are known.
    iii. Log loss function: In our project, we considered the cross-entropy or multi-class log loss function wherein we measure the distance between the output probability with the actual truth value. A good classifier would be the one whose output probabilities would be very near to the truth value. The cross-entropy loss is measure as:

    $$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_n^c \log(p_n^c)$$

    where N is the total number of samples (training/ testing), C is the total number of class (here in our case 12), p(n,c) is the probability of sample 'n' belonging to the 'c' class as the output for a selected classifier and y(n,c) is the true probability of sample 'n' belonging to 'c' class.

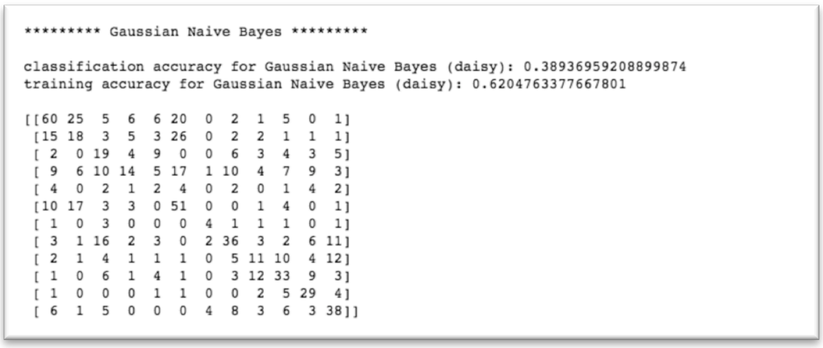
Hence, to select the two best classifiers, their accuracy scores on the validating dataset should be high and the log loss function should be low.

From the first stage of our experiment, we noted that the Naïve Bayes, K-NN and Decision Tree classifiers were the worst performing classifiers for all the given feature descriptors. We inferred that all these techniques grouped as hard classifiers were unsuccessful while predicting the classes for the validation dataset as they directly try to classify without estimating the other class conditional probabilities. These hard classifiers had as large as two-digit average log loss function clearly indicating that the distance between the output probability and the truth value is very large depicting them as bad classifiers irrespective of the feature descriptors selected. The log loss values for each classifier with different feature descriptors are mentioned below in the table. Also, as an example, all the three accuracy check parameters for all the five classifiers with the feature descriptor 'daisy' is shown below. (Please refer to 'Experiment - Level 1' jupyter notebook for rest of the feature descriptors)

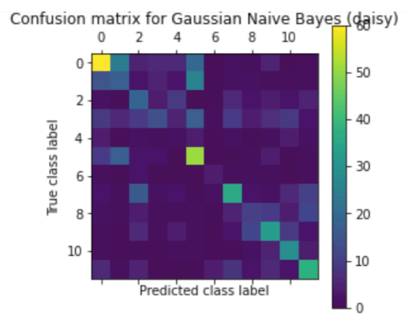
Classifier/ Feature Descriptor	Boost	Freak	Daisy	Lucid	ORB	SIFT	VGG
Naïve Bayes	21.12	21.49	20.35	27.74	19.76	18.22	19.57
KNN	22.23	23.22	16.13	22.8	20.93	14.73	22.52
Decision Tree	26.59	26	24.45	27.53	30.0	27.06	27.15
Random Forest	1.79	1.89	1.82	2.04	2.25	1.94	1.88
SVM	1.566	1.55	1.38	1.84	1.84	1.42	1.5

(Table 1: Average Log Loss value for all classifier for respective feature descriptors)

1. **Average Log Loss value for Naïve Bayes with daisy descriptor:**

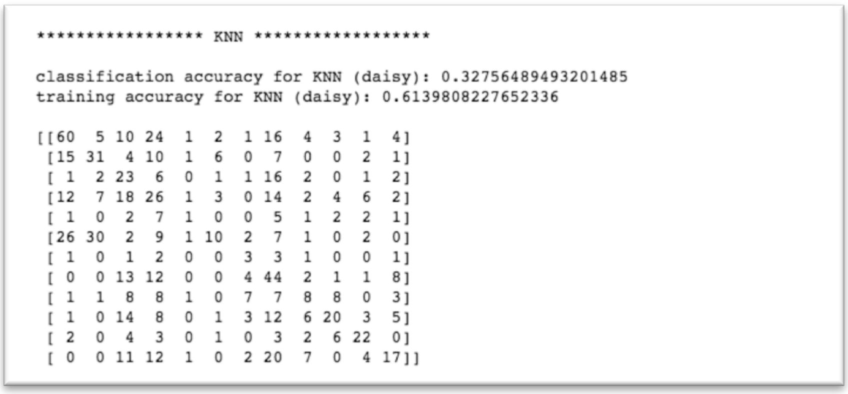


```
********* Gaussian Naive Bayes *********

classification accuracy for Gaussian Naive Bayes (daisy): 0.38936959208899874
training accuracy for Gaussian Naive Bayes (daisy): 0.6204763377667801

[[60 25  5  6  6 20  0  2  1  5  0  1]
 [15 18  3  5  3 26  0  2  2  1  1  1]
 [ 2  0 19  4  9  0  0  6  3  4  3  5]
 [ 9  6 10 14  5 17  1 10  4  7  9  3]
 [ 4  0  2  1  2  4  0  2  0  1  4  2]
 [10 17  3  3  0 51  0  0  1  4  0  1]
 [ 1  0  3  0  0  0  4  1  1  1  0  1]
 [ 3  1 16  2  3  0  2 36  3  2  6 11]
 [ 2  1  4  1  1  1  0  5 11 10  4 12]
 [ 1  0  6  1  4  1  0  3 12 33  9  3]
 [ 1  0  0  0  1  1  0  0  2  5 29  4]
 [ 6  1  5  0  0  0  4  8  3  6  3 38]]
```

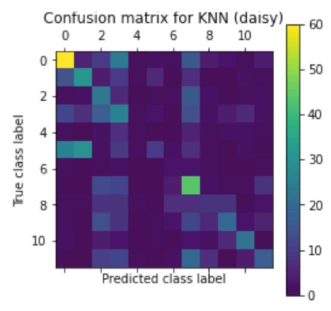


```
Average Log Loss for  Gaussian Naive Bayes  ( daisy ): 20.355
```

2. Average Log Loss value for KNN with daisy descriptor:

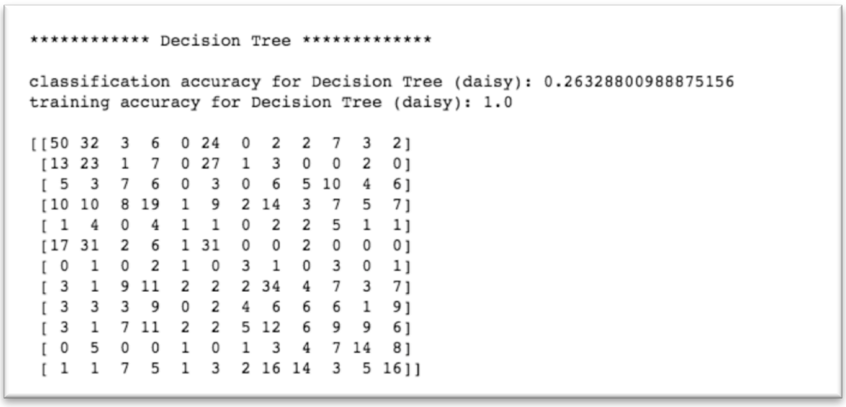


```
***************** KNN ******************

classification accuracy for KNN (daisy): 0.32756489493201485
training accuracy for KNN (daisy): 0.6139808227652336

[[60  5 10 24  1  2  1 16  4  3  1  4]
 [15 31  4 10  1  6  0  7  0  0  2  1]
 [ 1  2 23  6  0  1  1 16  2  0  1  2]
 [12  7 18 26  1  3  0 14  2  4  6  2]
 [ 1  0  2  7  1  0  0  5  1  2  2  1]
 [26 30  2  9  1 10  2  7  1  0  2  0]
 [ 1  0  1  2  0  0  3  3  1  0  0  1]
 [ 0  0 13 12  0  0  4 44  2  1  1  8]
 [ 1  1  8  8  1  0  7  7  8  8  0  3]
 [ 1  0 14  8  0  1  3 12  6 20  3  5]
 [ 2  0  4  3  0  1  0  3  2  6 22  0]
 [ 0  0 11 12  1  0  2 20  7  0  4 17]]
```

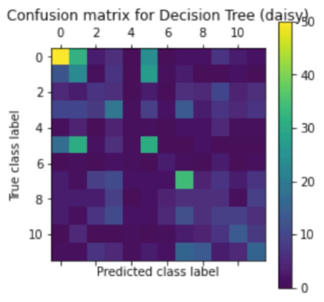


```
Average Log Loss for  KNN  ( daisy ): 16.131
```

3. Average Log Loss value for Decision Tree with daisy descriptor:

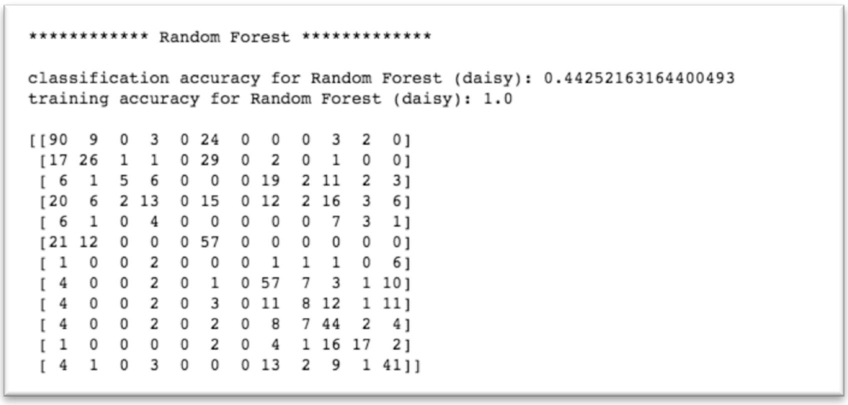


```
************ Decision Tree *************

classification accuracy for Decision Tree (daisy): 0.26328800988875156
training accuracy for Decision Tree (daisy): 1.0

[[50 32  3  6  0 24  0  2  2  7  3  2]
 [13 23  1  7  0 27  1  3  0  0  2  0]
 [ 5  3  7  6  0  3  0  6  5 10  4  6]
 [10 10  8 19  1  9  2 14  3  7  5  7]
 [ 1  4  0  4  1  1  0  2  2  5  1  1]
 [17 31  2  6  1 31  0  0  2  0  0  0]
 [ 0  1  0  2  1  0  3  1  0  3  0  1]
 [ 3  1  9 11  2  2  2 34  4  7  3  7]
 [ 3  3  3  9  0  2  4  6  6  6  1  9]
 [ 3  1  7 11  2  2  5 12  6  9  9  6]
 [ 0  5  0  0  1  0  1  3  4  7 14  8]
 [ 1  1  7  5  1  3  2 16 14  3  5 16]]
```

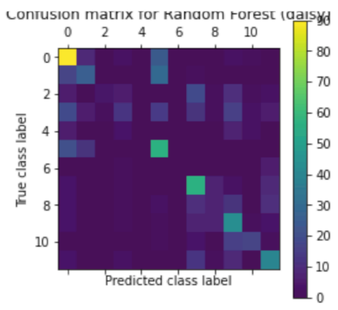


```
Average Log Loss for  Decision Tree  ( daisy ): 25.445
```

4. Average Log Loss value for Random Forest with daisy descriptor:

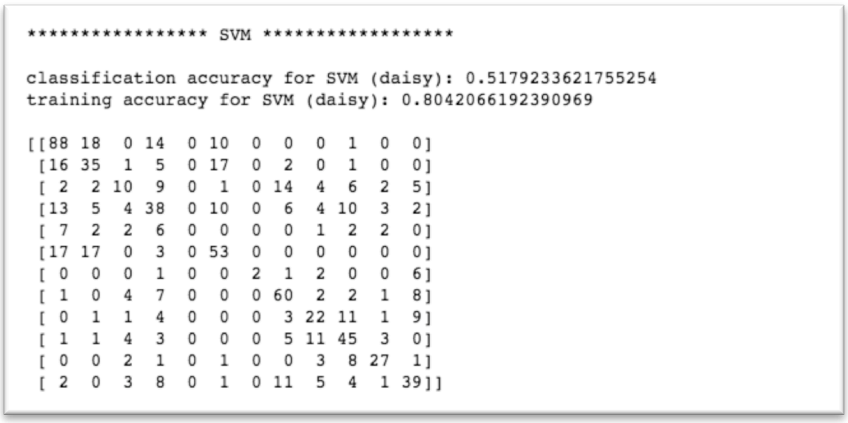


```
************ Random Forest *************

classification accuracy for Random Forest (daisy): 0.44252163164400493
training accuracy for Random Forest (daisy): 1.0

[[90  9  0  3  0 24  0  0  0  3  2  0]
 [17 26  1  1  0 29  0  2  0  1  0  0]
 [ 6  1  5  6  0  0  0 19  2 11  2  3]
 [20  6  2 13  0 15  0 12  2 16  3  6]
 [ 6  1  0  4  0  0  0  0  0  7  3  1]
 [21 12  0  0  0 57  0  0  0  0  0  0]
 [ 1  0  0  2  0  0  0  1  1  1  0  6]
 [ 4  0  0  2  0  1  0 57  7  3  1 10]
 [ 4  0  0  2  0  3  0 11  8 12  1 11]
 [ 4  0  0  2  0  2  0  8  7 44  2  4]
 [ 1  0  0  0  0  2  0  4  1 16 17  2]
 [ 4  1  0  3  0  0  0 13  2  9  1 41]]
```

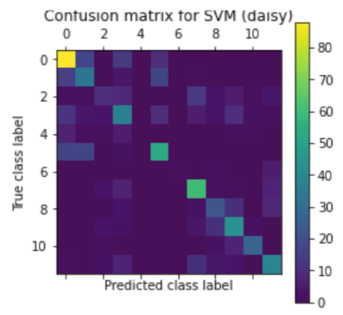


```
Average Log Loss for  Random Forest  ( daisy ): 1.828
```

5. **Average Log Loss value for SVM with daisy descriptor:**



```
***************** SVM ******************

classification accuracy for SVM (daisy): 0.5179233621755254
training accuracy for SVM (daisy): 0.8042066192390969

[[88 18  0 14  0 10  0  0  0  1  0  0]
 [16 35  1  5  0 17  0  2  0  1  0  0]
 [ 2  2 10  9  0  1  0 14  4  6  2  5]
 [13  5  4 38  0 10  0  6  4 10  3  2]
 [ 7  2  2  6  0  0  0  0  1  2  2  0]
 [17 17  0  3  0 53  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  2  1  2  0  0  6]
 [ 1  0  4  7  0  0  0 60  2  2  1  8]
 [ 0  1  1  4  0  0  0  3 22 11  1  9]
 [ 1  1  4  3  0  0  0  5 11 45  3  0]
 [ 0  0  2  1  0  1  0  0  3  8 27  1]
 [ 2  0  3  8  0  1  0 11  5  4  1 39]]
```



```
Average Log Loss for  SVM  ( daisy ): 1.389
```

2.3 Experiment: Level 2

From level 1, we discarded the three hard classifiers and for the remaining two namely, Random Forest and SVM, we then observed their respective accuracy score (mentioned in the Table 2) and average log loss value (mentioned in Table 1) for all the seven feature descriptors.

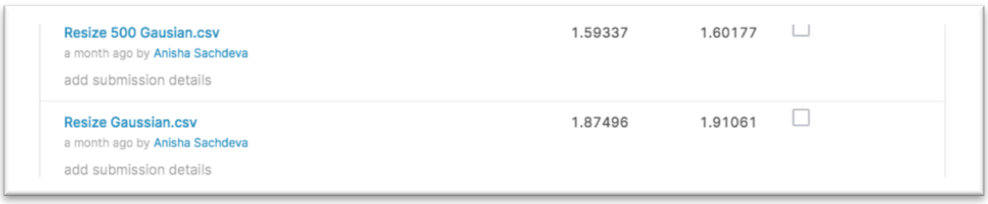
Classifier/Feature Descriptor	Boost	Freak	Daisy	Lucid	ORB	SIFT	VGG
SVM	0.44	0.40	0.51	0.38	0.36	0.50	0.47
Random Forest	0.36	0.38	0.44	0.37	0.26	0.37	0.38

(Table 2: Accuracy Score for SVM and Random classifier for seven feature descriptors)

'Daisy' & 'SIFT' feature descriptors had the highest accuracy score and lowest log loss values for both the classifiers SVM & Random Forest. We selected these two classifiers and features and tried to improve their accuracy by performing some pre-processing techniques.

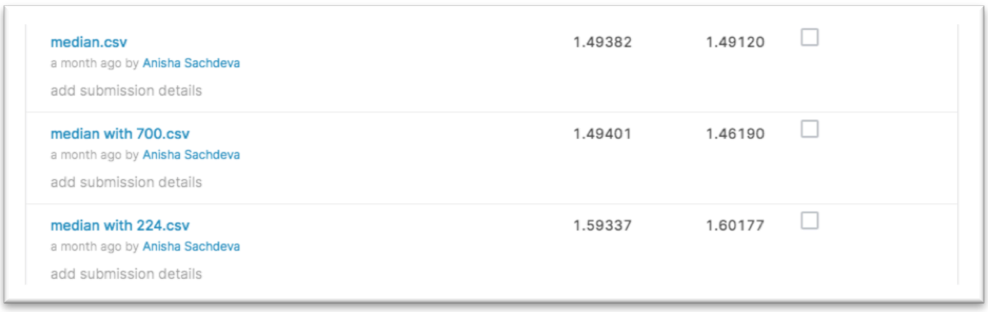
1. **Image pre-processing:** As our raw images were of different sizes, to have the same aspect ratios of all the images and have a base size, we resized the images in the training dataset. The resize parameter was selected as 500*500 as we thought this would be an ideal size for all the images, neither too large nor too small and they all would be rescaled to square shape. Further, to remove the noise from the data we applied the gaussian blur technique with radius set as 5 which basically intensifies an image at the center and the intensity diminishes for pixel which move away from the center. This helps to supress the high frequency components and in turn

noise by generating a blurred image. *(Before finalizing 500 as the resize parameter, we tried with the value of parameter as 224 along with gaussian blur technique, however, 500*500 aspect ratio gave us better results on the Kaggle competition website and hence 500 as the parameter was selected Further, we also tried the median filter technique without resizing as well as with resizing of image with parameter once set as 224 and the other time set as 700, but the scores on Kaggle were worst for median technique with 224*224 resized image followed by median filter without any resizing of image and at last median filter with 700*700 aspect ratio. All these experiments whose Kaggle score is mentioned below were performed with daisy descriptor and SVM classifier as this combination had provided us with the best results in level 1. We did not dig down with such other combinations of resizing and blurring techniques as none of them could improvise the model)*



Resize 500 Gausian.csv
a month ago by Anisha Sachdeva
add submission details
1.59337 1.60177

Resize Gaussian.csv
a month ago by Anisha Sachdeva
add submission details
1.87496 1.91061



median.csv
a month ago by Anisha Sachdeva
add submission details
1.49382 1.49120

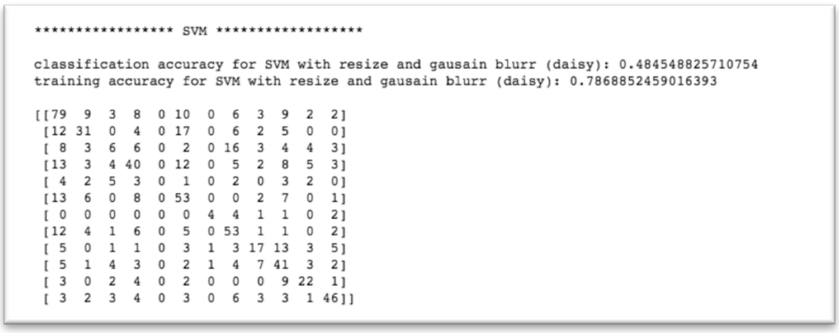
median with 700.csv
a month ago by Anisha Sachdeva
add submission details
1.49401 1.46190

median with 224.csv
a month ago by Anisha Sachdeva
add submission details
1.59337 1.60177

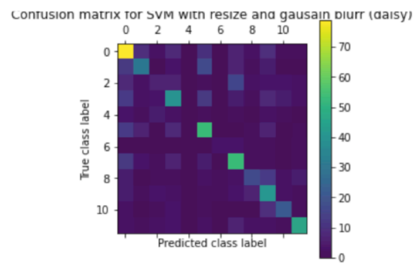
2. **Training data split:** As done in level 1, we again split out training data in the ratio of 80:20. 3233 images were trained and 809 images were validated.
3. **Feature extraction & training the model:** Now, for the new training and validation datasets with the pre-processed images, we extracted the Daisy and SIFT feature descriptors and using the K-means clustering algorithm, clustered the features with the size of codebook set as 1700. Further, using the newly generated clusters we trained our two models: SVM and Random Forest.
4. **Checking the accuracy of the model:** We observed that after resizing and applying gaussian blur, the accuracy score for both the classifiers decreased and the average log loss value increased meaning that applying both the pre-processing techniques could not improve the performance of both the classifiers. The possible reason for the resizing technique being failed could be for some very small size of images, some pixels would have been distorted while enlarging them to the aspect ratio of 500*500.
   However, we can conclude from this part of the experiment that in any condition that has been applied so far, SVM classifier with Daisy feature descriptor outperforms all the other classifiers. As an example, all the three accuracy check parameters for SVM and Random Forest with the feature descriptor 'daisy' is shown below. (Please refer to 'Experiment - Level 2' jupyter notebook for rest of the SIFT feature descriptor)

Classifier/Feature Descriptor	Daisy		SIFT	
	Accuracy Score	Log Loss Value	Accuracy Score	Log Loss Value
SVM	0.48	1.59	0.42	1.69
Random Forest	0.37	2.07	0.30	2.22

1. **Accuracy parameters for SVM with daisy descriptor:**

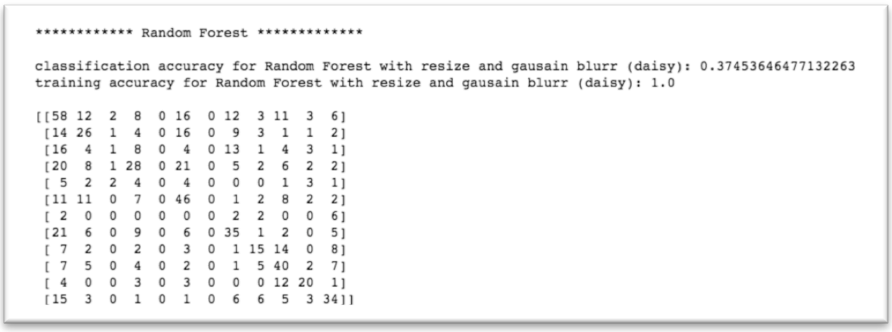


```
***************** SVM ******************

classification accuracy for SVM with resize and gausain blurr (daisy): 0.484548825710754
training accuracy for SVM with resize and gausain blurr (daisy): 0.7868852459016393

[[79  9  3  8  0 10  0  6  3  9  2  2]
 [12 31  0  4  0 17  0  6  2  5  0  0]
 [ 8  3  6  6  0  2  0 16  3  4  4  3]
 [13  3  4 40  0 12  0  5  2  8  5  3]
 [ 4  2  5  3  0  1  0  2  0  3  2  0]
 [13  6  0  8  0 53  0  0  2  7  0  1]
 [ 0  0  0  0  0  0  4  4  1  1  0  2]
 [12  4  1  6  0  5  0 53  1  1  0  2]
 [ 5  0  1  1  0  3  1  3 17 13  3  5]
 [ 5  1  4  3  0  2  1  4  7 41  3  2]
 [ 3  0  2  4  0  2  0  0  0  9 22  1]
 [ 3  2  3  4  0  3  0  6  3  3  1 46]]
```

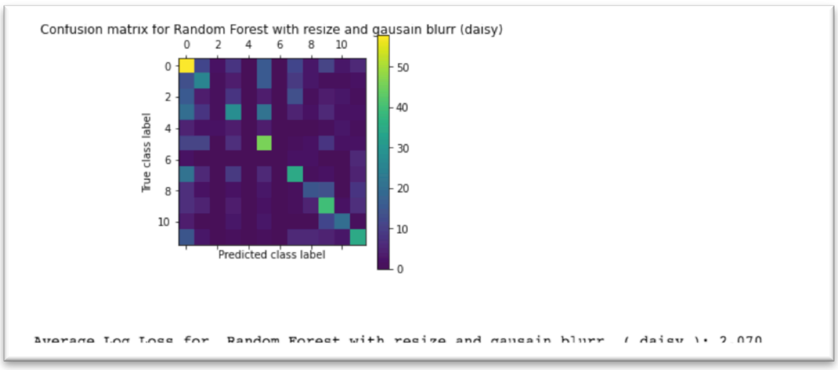


```
Average Log Loss for  SVM with resize and gausain blurr  ( daisy ): 1.599
```

2. **Accuracy parameters for random forest with daisy descriptor:**



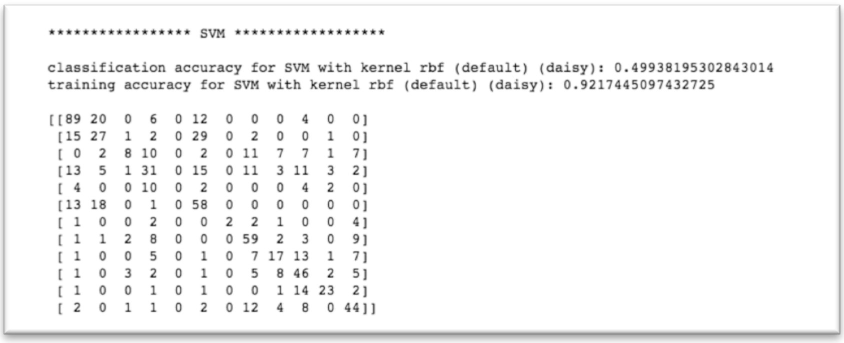
```
************ Random Forest *************

classification accuracy for Random Forest with resize and gausain blurr (daisy): 0.37453646477132263
training accuracy for Random Forest with resize and gausain blurr (daisy): 1.0

[[58 12  2  8  0 16  0 12  3 11  3  6]
 [14 26  1  4  0 16  0  9  3  1  1  2]
 [16  4  1  8  0  4  0 13  1  4  3  1]
 [20  8  1 28  0 21  0  5  2  6  2  2]
 [ 5  2  2  4  0  4  0  0  0  1  3  1]
 [11 11  0  7  0 46  0  1  2  8  2  2]
 [ 2  0  0  0  0  0  0  2  2  0  0  6]
 [21  6  0  9  0  6  0 35  1  2  0  5]
 [ 7  2  0  2  0  3  0  1 15 14  0  8]
 [ 7  5  0  4  0  2  0  1  5 40  2  7]
 [ 4  0  0  3  0  3  0  0  0 12 20  1]
 [15  3  0  1  0  1  0  6  6  5  3 34]]
```



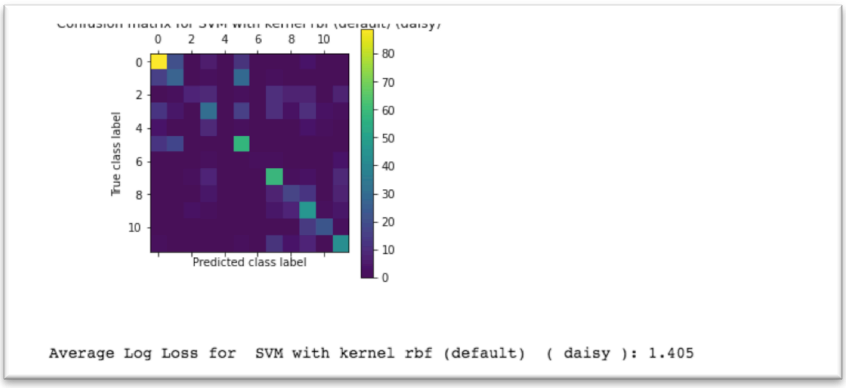
```
Average Log Loss for  Random Forest with resize and gausain blurr  ( daisy ): 2.070
```

Special Mention: While in level 2, we also tried to fine tune the hyper-parameter ‘kernel’ of SVM classification algorithm. We read from the scikit-learn.org, that kernel parameter could have five possible values as {‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or default=’rbf’}. In order to find the best suited value, we applied all these possible values one by one while defining our SVM classifier with daisy descriptor and noted the accuracy score and log loss with each of the parameter value. We noted that best suited value for ‘kernel’ was ‘linear’ as it produced the highest accuracy and lowest log loss. Hence, for all our experiments we defined the SVM classifier with kernel parameter as linear. As an example, below images depict the accuracy parameters found for the default ‘rbf’ and ‘linear’ value of the parameter ‘kernel’. (For rest of the value, please refer to ‘Optimizing Kernel - SVM hyper parameter’ jupyter notebook)

1. **Accuracy parameters for default kernel value in SVM**

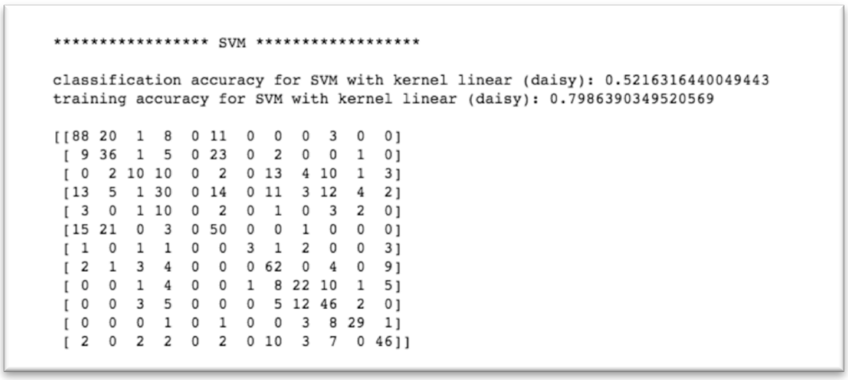


```
***************** SVM ******************

classification accuracy for SVM with kernel rbf (default) (daisy): 0.49938195302843014
training accuracy for SVM with kernel rbf (default) (daisy): 0.9217445097432725

[[89 20  0  6  0 12  0  0  0  4  0  0]
 [15 27  1  2  0 29  0  2  0  0  1  0]
 [ 0  2  8 10  0  2  0 11  7  7  1  7]
 [13  5  1 31  0 15  0 11  3 11  3  2]
 [ 4  0  0 10  0  2  0  0  0  4  2  0]
 [13 18  0  1  0 58  0  0  0  0  0  0]
 [ 1  0  0  2  0  0  2  2  1  0  0  4]
 [ 1  1  2  8  0  0  0 59  2  3  0  9]
 [ 1  0  0  5  0  1  0  7 17 13  1  7]
 [ 1  0  3  2  0  1  0  5  8 46  2  5]
 [ 1  0  0  1  0  1  0  0  1 14 23  2]
 [ 2  0  1  1  0  2  0 12  4  8  0 44]]
```

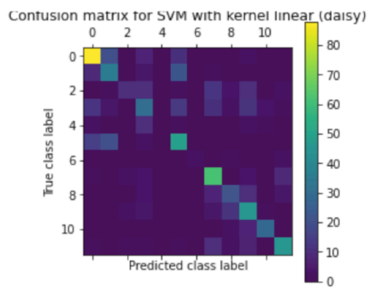


```
Average Log Loss for  SVM with kernel rbf (default)  ( daisy ): 1.405
```

2. Accuracy parameters for kernel value as ‘linear’ in SVM



```
***************** SVM ******************

classification accuracy for SVM with kernel linear (daisy): 0.5216316440049443
training accuracy for SVM with kernel linear (daisy): 0.7986390349520569

[[88 20  1  8  0 11  0  0  0  3  0  0]
 [ 9 36  1  5  0 23  0  2  0  0  1  0]
 [ 0  2 10 10  0  2  0 13  4 10  1  3]
 [13  5  1 30  0 14  0 11  3 12  4  2]
 [ 3  0  1 10  0  2  0  1  0  3  2  0]
 [15 21  0  3  0 50  0  0  1  0  0  0]
 [ 1  0  1  1  0  0  3  1  2  0  0  3]
 [ 2  1  3  4  0  0  0 62  0  4  0  9]
 [ 0  0  1  4  0  0  1  8 22 10  1  5]
 [ 0  0  3  5  0  0  0  5 12 46  2  0]
 [ 0  0  0  1  0  1  0  0  3  8 29  1]
 [ 2  0  2  2  0  2  0 10  3  7  0 46]]
```



```
Average Log Loss for  SVM with kernel linear  ( daisy ): 1.370
```

Experiment: Level 3

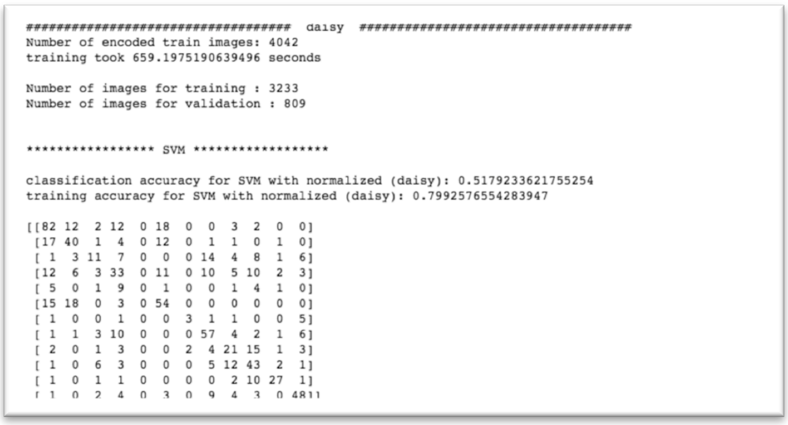
Now, we only focused on the SVM classifier with daisy feature descriptor and attempted to improve the performance of our model.

1. **Image pre-processing:** We normalized the images present in the training dataset implying that we changed the pixel intensity to improvise the contrast in images present in the training dataset. The default parameters for normalization were considered.
2. **Training data split:** Once again the data was split into the ratio of 80:20. 3233 images were trained and 809 images were validated.
3. **Feature extraction & training the model:** Using the normalized image training dataset (containing both training and validation sets), we extracted the ‘daisy’ feature and clustered it using the codebook size as 1700. Again, we trained our SVM model using the new set of features and tested its performance on the validation dataset using the three parameters as mentioned above.
   Further, we explored about the bagging classifier, a meta-estimator which we used with our SVM classifier to create random subsets of the training dataset and then the individual predictions were aggregated (by averaging or voting) in order to form the final prediction. We trained the SVM bagging classifier and further tested its performance.
4. **Checking the accuracy of the model:** We noted that our model somewhat has almost equivalent scores with raw data, normalized data and normalized data with bagging technique. There is a very slight difference wherein normalized data with bagging technique performs little better but still it is not a very significant difference. All the three SVM model almost have the similar accuracy score as 0.51 and average log loss value ranging between 1.35 to 1.39.

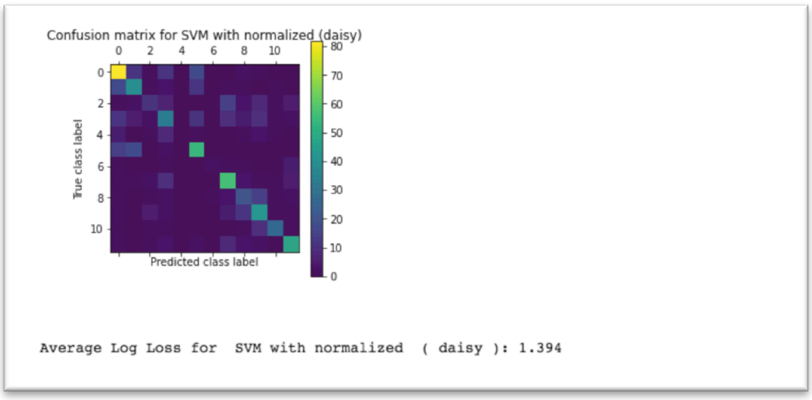
All the three accuracy check parameters for normalized SVM and normalized SVM bagging technique with the feature descriptor 'daisy' is shown below.

Classifier/Feature Descriptor	Normalization		Normalization + Bagging	
	Accuracy Score	Log Loss Value	Accuracy Score	Log Loss Value
SVM	0.51	1.39	0.52	1.35

1. **Accuracy parameters for Normalized SVM with daisy descriptor:**

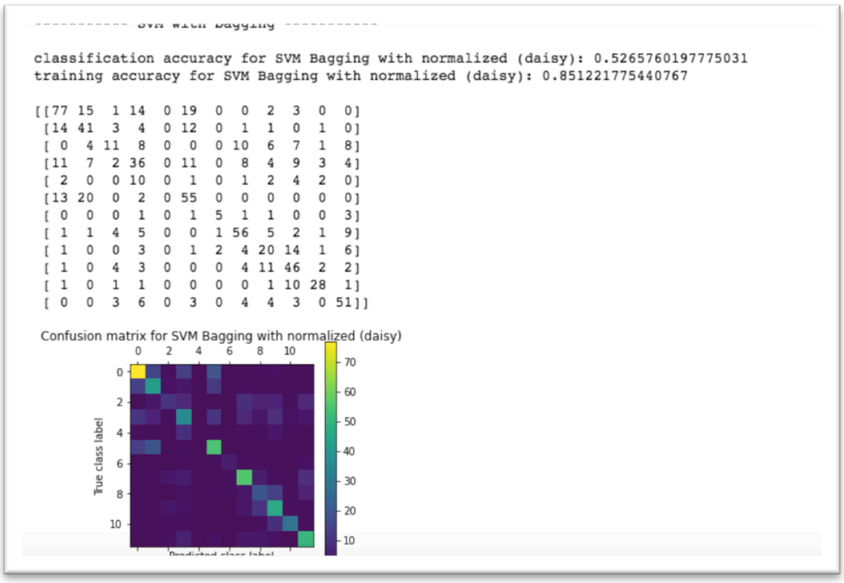


```
###################################  daisy  ####################################
Number of encoded train images: 4042
training took 659.1975190639496 seconds

Number of images for training : 3233
Number of images for validation : 809


***************** SVM ******************

classification accuracy for SVM with normalized (daisy): 0.5179233621755254
training accuracy for SVM with normalized (daisy): 0.7992576554283947

[[82 12  2 12  0 18  0  0  3  2  0  0]
 [17 40  1  4  0 12  0  1  1  0  1  0]
 [ 1  3 11  7  0  0  0 14  4  8  1  6]
 [12  6  3 33  0 11  0 10  5 10  2  3]
 [ 5  0  1  9  0  1  0  0  1  4  1  0]
 [15 18  0  3  0 54  0  0  0  0  0  0]
 [ 1  0  0  1  0  0  3  1  1  0  0  5]
 [ 1  1  3 10  0  0  0 57  4  2  1  6]
 [ 2  0  1  3  0  0  2  4 21 15  1  3]
 [ 1  0  6  3  0  0  0  5 12 43  2  1]
 [ 1  0  1  1  0  0  0  0  2 10 27  1]
 [ 1  0  2  4  0  3  0  9  4  3  0 48]]
```



```
Average Log Loss for  SVM with normalized  ( daisy ): 1.394
```

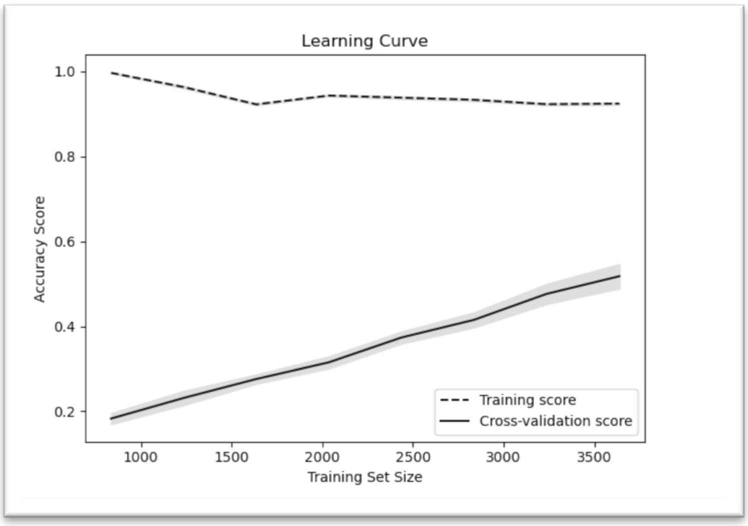
2. Accuracy parameters for Normalized SVM bagging technique with daisy descriptor:

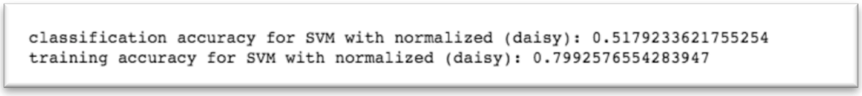


```
----------- SVM with Bagging -----------

classification accuracy for SVM Bagging with normalized (daisy): 0.5265760197775031
training accuracy for SVM Bagging with normalized (daisy): 0.851221775440767

[[77 15  1 14  0 19  0  0  2  3  0  0]
 [14 41  3  4  0 12  0  1  1  0  1  0]
 [ 0  4 11  8  0  0  0 10  6  7  1  8]
 [11  7  2 36  0 11  0  8  4  9  3  4]
 [ 2  0  0 10  0  1  0  1  2  4  2  0]
 [13 20  0  2  0 55  0  0  0  0  0  0]
 [ 0  0  0  1  0  1  5  1  1  0  0  3]
 [ 1  1  4  5  0  0  1 56  5  2  1  9]
 [ 1  0  0  3  0  1  2  4 20 14  1  6]
 [ 1  0  4  3  0  0  0  4 11 46  2  2]
 [ 1  0  1  1  0  0  0  0  1 10 28  1]
 [ 0  0  3  6  0  3  0  4  4  3  0 51]]
```

3. Result

Finally, now we were exhausted of all the commonly known pre-processing techniques with various combinations of well-known feature descriptors for different classifiers, we selected our best model as the normalized set of images for the SVM classifier with 'daisy' descriptor. To verify, if our model was overfitted or underfitted, we extracted the learning curve.





```
classification accuracy for SVM with normalized (daisy): 0.5179233621755254
training accuracy for SVM with normalized (daisy): 0.7992576554283947
```

We observe that there is a huge gap between the training and cross validation score implying high variance i.e., our model is overfitted. The model fits the training data too well but provides poor performance while classification. This result can also be verified with the training accuracy being almost 80% while the classification accuracy is only 51.7%.

Kaggle Score: The same model was used for the test dataset and the csv file containing the probabilities of all images belonging to the 12 classes were extracted and uploaded on the Kaggle leaderboard. The final accuracy score on learderboard was achieve as 1.36, corresponding to the 8th position out of 33 contenders.