VRIJE
UNIVERSITEIT
BRUSSEL

FINAL REPORT

Flower Classification

Mengyao Song 0563486

August 20, 2021

Sciences and Bio-Engineering Sciences

1 Introduction

Predicting flowers species by traditional machine learning models is a challenging project. When it comes to image classification, CNN(Convolution Neural Network) model is widely used in the industry, however, the complexity of CNN may overwhelms the machine learning beginners. I use the traditional machine learning methods in the project because they offer a learning path for those who are new to the core concepts. In order to properly predict the flowers, this project contains data preprocessing, several procedures used for model assessment and selection, hyperparameter optimization and evaluation. My attempt can be divided into two parts: fit the models without bag-of-words, fit the models with bag-of-words.

2 Data preprocessing

In the part of *bag of visual words*, i simply resize the images after read them. Scale-invariant feature transform, or *SIFT*, is used here to detect and compute keypoints and descriptors. It can generates large numbers of features that densely cover the image over the full range scales and locations. It also performs well when the images are different both in scales and rotations. Train the KMeans classifier with the descriptors and extract the features by the function *extractFeatures*. I normalize the features by *scale.transform()*. The frequency of the extracted visual word is drawn below:

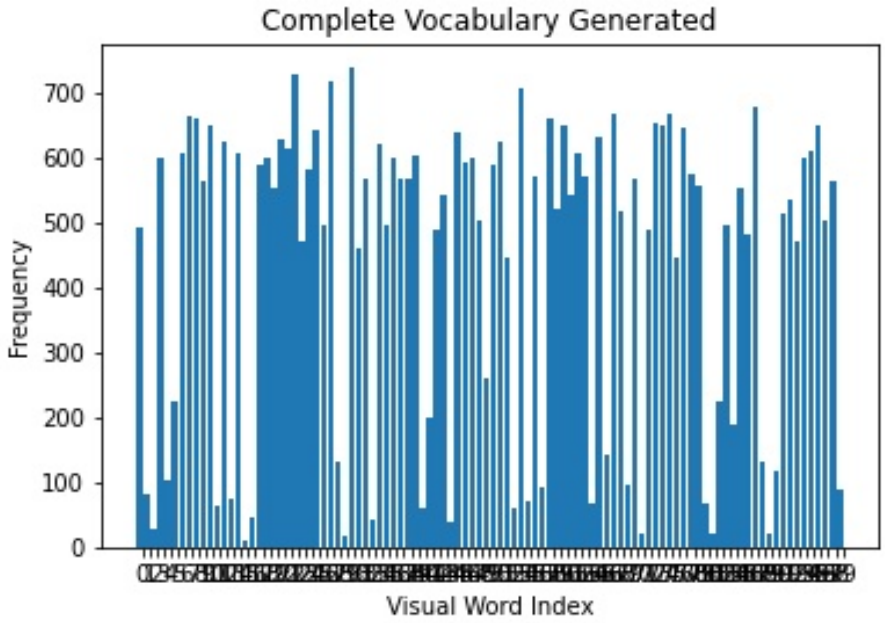

Figure 1: Vocabulary Generation

In the part that without bag-of-words, a series of preprocessing is adopted on the images.

- use BGR2GRAY to convert images to gray image.
- scale gray images by the function *normalizeGrayscale(images)*. It is a min-max scaling that the values end up ranging from 0 to 1.
- resize the images into 220*220 and flatten them.

In both two parts, labels are encoded by *LabelEncoder()* that encodes target labels with value between 0 and n_classes-1. The processed datasets are divided into training set and validation set as the ratio of 0.8. I train my models using the training set, and test it using the validation set.

3 Model selection

As I declared in the previous part, the model selection is also divided into two main parts. In the section 3.1, Support Vector Classification with bag of visual is explained here. In the rest sections, the techniques without BOVW are explained.

Since the fits are scored using the multi-class Log Loss, to evaluate the models, I implemented a function names *categorical_cross_entropy(actual, predicted)*. The formula of the multi-class Log Loss is:

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

where N is the number of training/test samples, C is the number of classes and p_n^c is the probability that sample n belongs to class c as output of your classifier. y_n^c is the true class probability that sample n belongs to class c. $y_n^c = 1$ if sample n belongs to class c and $y_n^c = 0$ otherwise.

According to the formula, the loss is calculated as follow:
mean_sum_score = - 1.0 / len(actual) *(y[i][j] * log(1e-15 + predicted[i][j]))

3.1 Bag-of-Visual$_W ordsandSupportVectorClassification$

The bag-of-visual-words model can be applied to image classification by treating image features as words. It is a vector of occurrence counts of a vocabulary of image features. Roughly speaking, it requires the following steps:

- Feature representation. A good descriptor should have the ability to handle intensity, rotation and scale. Here, as mentioned previous, Sift is used to provide support.
- Codebook generation. The final step for the BoVW model is to convert vector-represented patches to "codewords". In the project, k-means clustering is performed over the vectors.

After the codebook is generated, I use support vector classifier to do the classification. Support vector classification, or *svc*, is one of the support vector machines. It is based on libsvm. *Svc* is handled according to one-vs-one scheme when there are multi-classes.

In the project, I adopt the *precomputed* kernel. The parameter C and *gammas* are tuned by *RandomizedSearchCV*. Instead of trying out all possible combinations, it evaluates a given number of random combinations for each hyperparameter at every iteration. The search grid is shown below:

- Cs = [0.5, 0.1, 0.15, 0.2, 0.3]
- gammas = [0.1, 0.11, 0.095, 0.105]

After searching, C with value 0.15 and gamma with value 0.105 are selected. Train *svc* with the training data and test it on the validation data. The prediction of the validation set is implemented by *svm.predict_proba* so that I can get a series probabilities for the target.

For the support vector classification, the cross-entropy loss is 1.726383742973774.

3.2 K-NearestNeighbor classifier

The next model we're going to try is created using k-Nearest Neighbor. The KNN classifier is one of the most simple image classification algorithm. It relies on the distance between feature vectors.

The search grid is shown below:

- "n_neighbors": [1,5,7,9,11,13,15,17,19,21,23,25,30,50,100,200,500],
- "metric": ["euclidean", "cityblock"]

[INFO] grid search accuracy: 29.11%
[INFO] grid search best parameters: 'metric': 'cityblock', 'n_neighbors': 15

Each entry will be classified based on the cityblock distances using the selected features and their dimensions. The city block distance is defined as the following formula:

$$d(p, q) = \sum_{i-1}^{N} |q_i - p_i|$$

The accuracy of KNN is only 27.95%. To observe the results in more detail, I draw the heatmap of the prediction to compare with the true result. The heatmap is shown below:

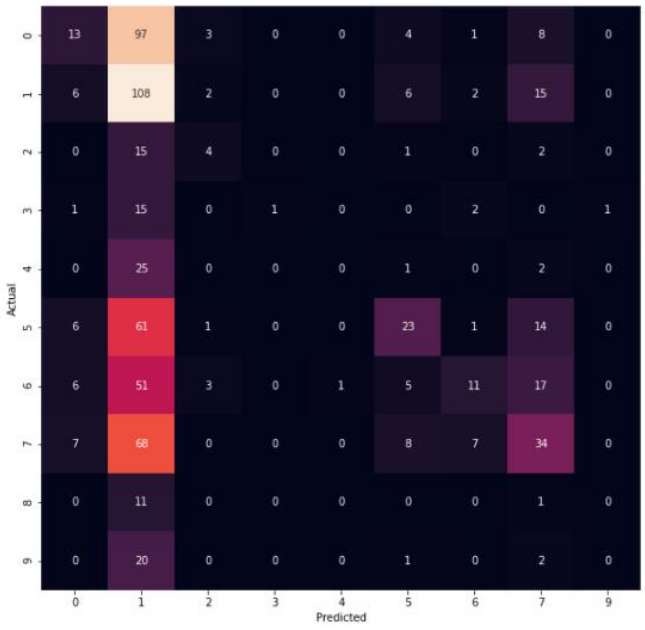

Figure 2: The heatmap of knn

It could be seen that most of the images are classified into the second species, and none of the images is classified as the ninth species. The classifier is not reliable according to the result of the validation.

3.3 Logistic Regression

Logistic regression is very popular in machine learning. It can work on both binary and multi-class classification very well.

- 'penalty':['none'],
- 'tol':[0.0001, 0.1],
- 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
- 'multi_class' : ['multinomial'],
- 'penalty' : ['l2'],
- 'solver' : ['newton-cg', 'lbfgs', 'sag', 'saga'],
- 'max_iter': list(range(100,600,100))

[INFO] grid search accuracy: 24.78%

[INFO] grid search best parameters: 'tol': 0.0001, 'solver': 'newton-cg', 'penalty': 'l2', 'multi_class': 'multinomial', 'max_iter': 300, 'C': 0.01

The heatmap is shown in Figure 3. The main problem of the logistic regression classifier is the same as the KNN that the flowers in species 3,4,5,9,10 can not be classified properly.

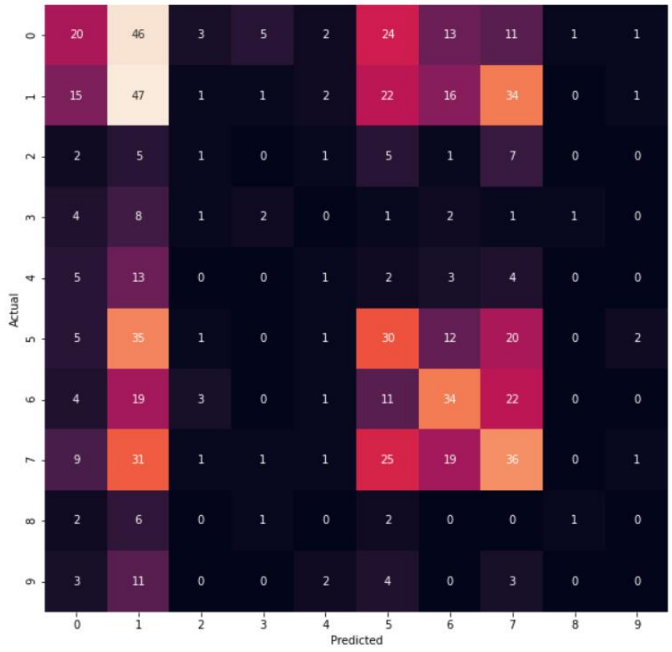

Figure 3: The heatmap of logistic regression

When I studied the logistic regression classifier further, I realized why is the accuracy of the classifier so low. What I did not take into account is that logistic regression becomes a classification technique only when a decision threshold is brought into the picture. Obviously in the project, the preprocessing is insufficient to provide the data needed for logistic regression classifier.

3.4 Random Forest classifier

Random forests is a supervised learning algorithm. It can be used both for classification and regression. Random forests are known to provide a high accuracy, even with a large amount of features. Here I use RandomizedSearchCV to select best parameters because I set a plenty of parameters. If GridSearchCV is adopted, it will definitely take a lot of time in computation.

- 'bootstrap': [True, False],
- 'max_depth': [10, 50, 100, None],
- 'max_features': ['auto', 'sqrt'],
- 'multi_class' : ['multinomial'],
- 'min_samples_leaf': [1, 4],
- 'min_samples_split': [2, 5, 10],
- 'n_estimators': [200, 400, 800, 1000,1500, 2000]

The result of this selection is:
[INFO] random search accuracy: 37.03%
[INFO] random search best parameters: 'n_estimators': 1500, 'min_samples_split': 2, 'min_samples_leaf': 4, 'max_features': 'auto', 'max_depth': None, 'bootstrap': False
Evaluate the rf classifier with the heatmap and categorical_cross_entropy, I get the following result:
The cross-entropy loss of random forest is: 1.7237461261059628

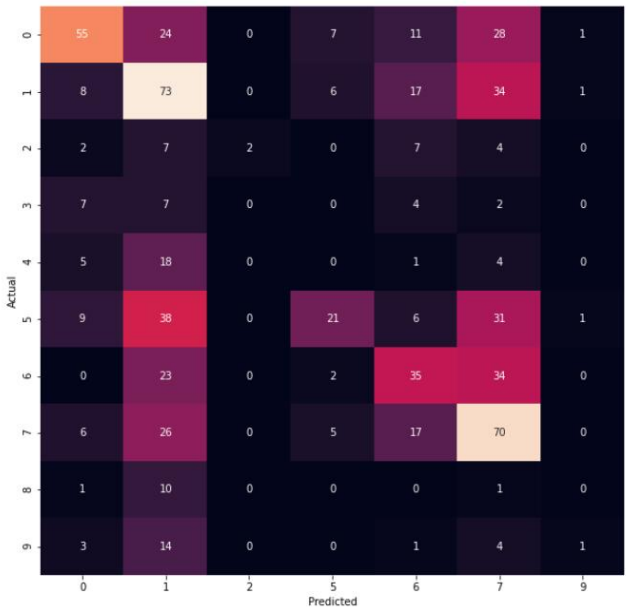

Figure 4: The heatmap of logistic regression

Relatively speaking, with my prepossessed dataset, the random forest classifier performs better than the previous two classifiers.

4 Conclusion

The project offers me a great chance to have hands-on experience in machine learning. I believe studying these models offers me a solid starting point for further study of advanced methods. By implementing different classifiers using two methods, I find that the bag-of-visual-words has significantly higher accuracy. Even though I did a lot in preprocessing and parameters tuning, the accuracy of the three classifiers which do not use BoVW is still lower than 40%. In the project, I do not cover cross validation and pruning techniques. It is worth discovering them in my further study.