ECSE-415 Final Project: Flower Classification & Segmentation

Durham Abric   Matthew Caminiti   Thomas Hillyer[1]

May 2, 2020

[1]Thank you for an interesting semester of computer vision!

Abstract

This report outlines the approach taken and the results obtained by Group 13 during the ECSE-415 (Computer Vision) final project in Winter 2020. First, we cover the classification task and discuss the implementation of the Support Vector Machine (SVM) as well as how we determined which feature selection technique to use. Next we discuss segmentation using a Gaussian Mixture Model (GMM) and the Normalized Graph Cut technique and how the hyperparameters for these methods were chosen. For both parts, we provide example outputs to compare with the ground truths provided by the course instructors.

Contents

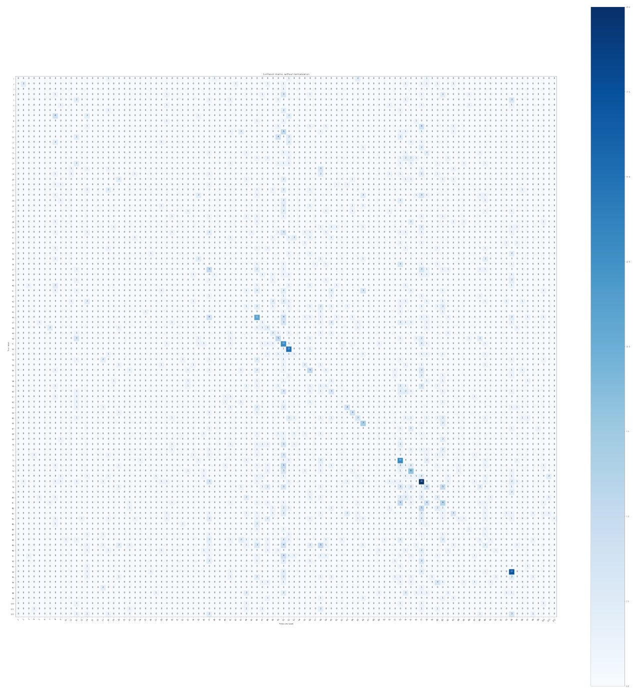
1 Classification **2**
1.1 Classification Results . . . 2
1.1.1 Confusion Matrix . . . 2
1.1.2 Validation Results . . . 2
1.2 Classification Methodology . . . 2
1.2.1 Dataset Description . . . 2
1.2.2 Feature Extraction . . . 3
1.2.3 Cross Validation & Hyperparameter Tuning . . . 4
1.3 Classification Discussion . . . 4
1.3.1 Performance Evaluation . . . 4
1.3.2 Intricacies of Classification Methodology . . . 4
1.4 Classification Examples . . . 4

2 Segmentation **7**
2.1 Segmentation Results . . . 7
2.1.1 Confusion Matrix . . . 7
2.1.2 DICE *(Overall)* . . . 7
2.1.3 DICE *(Per Validation Set)* . . . 8
2.2 Segmentation Methodology . . . 8
2.2.1 Image Preprocessing . . . 8
2.2.2 Segmentation Approaches . . . 8
2.2.3 Hyperparameter Selection . . . 9
2.2.4 Cross Validation . . . 9
2.3 Segmentation Discussion . . . 9
2.3.1 Performance Evaluation . . . 9
2.3.2 Methodology Strengths & Weaknesses . . . 10
2.4 Segmentation Examples . . . 10

1 Classification

1.1 Classification Results

1.1.1 Confusion Matrix



Validation Confusion Matrix

As the numbers are nearly indistinguishable from the confusion matrix, I will not reference the image, but instead directly discuss the numbers. Categories we had great difficulty with included but were not limited to: 79, 82, 90, 91. For these categories our classifier never correctly identified any, and our assumption is that the unique features for these categories were often overrun with clutter in the background.

1.1.2 Validation Results

The following table shows our weighted averages of our model's accuracy, precision, and recall for the 5 folds of our cross validation. Due to the minor variation in precision and recall from our accuracy, we cannot make any actionable insights to the dataset. In the event that our precision and recall numbers are significantly higher than our accuracy, it could suggest that certain samples in the dataset corresponding to a portion of the categories are poor samples.

1.2 Classification Methodology

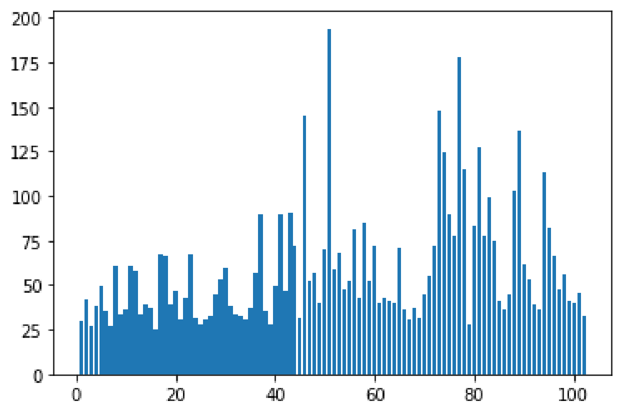
1.2.1 Dataset Description

The dataset used for the classification task of this project was a set of 6000 training images with 102 categories of flowers. Every flower image was in colour and whose dimensions were relatively close to 500px by 500px. On average, the dataset contained 58 training images per category, with

	Accuracy	Precision	Recall
Support Vector Machine	0.16	0.15	0.16
	0.15	0.11	0.15
	0.15	0.12	0.15
	0.15	0.13	0.15
	0.16	0.12	0.16
	St. Dev.	**St. Dev.**	**St. Dev.**
	0.0055	0.0083	0.0055

	Accuracy	Precision	Recall
Multinomial Naive Bayes	0.15	0.11	0.15
	0.137	0.09	0.14
	0.12	0.11	0.12
	0.132	0.09	0.13
	0.155	0.12	0.15
	St. Dev.	**St. Dev.**	**St. Dev.**
	0.013	0.012	0.012

variations in scale, lighting, orientation and depth. The breakdown of training samples per category is shown below.



Samples per Category

1.2.2 Feature Extraction

Our strategy for feature extraction for the classification task was creating a visual Bag of Words (BoW) through a K-means clustering of SIFT descriptors. Given the nature of the contents of the training images and the great differences in colour and shape of the flowers, we believed constructing a bag of visual words would yield great results for classification.

As our chosen method of feature extraction was SIFT, there was no preprocessing required for the images (there are some preprocessing steps in our codebase, but those were for our attempts at feature extraction using histograms of oriented gradients). The number of SIFT features to be used was pivotal to classification performance and computational complexity. SIFT descriptors are a constant 128-dimension vector and it is to our discretion as to how many we use. The decisions made will be further discussed in **Section 1.3.2**. Following our SIFT feature extraction we performed K-Means clustering on all descriptors discovered for every image in the training set. With our bag of visual words constructed, we then performed the final task of predicting which cluster each descriptor belonged to for every training image. The resultant set of features would be 6000 histograms whose dimensionality is the number of clusters used for K-Means clustering.

1.2.3 Cross Validation & Hyperparameter Tuning

As directed in the projected description, we performed k-fold cross validation with a Support Vector Machine. For our k-fold cross validation, we used 5 folds, in which 4800 training samples would be used to fit our classifier, and the remaining 1200 used for validation.

The hyperparameters subject to tuning included: number of best SIFT features, number of clusters for BoW, maximum iterations for SVM, regularization parameter of SVM, and the tolerance. For the hyperparameters specific to the SVM, these were tuned in conjunction with our 5-fold cross validation to determine those with the greatest sucess. The number of SIFT features and number of clusters were ultimately decided by our available computing resources. Engaging with more than 50 SIFT features per image and over 200 cluster centers resulted in egregiously long run times and higher values could result in overuse of memory and computer shut down. Within the boundaries of our computing power, we found that using the 20 best SIFT features along with 300 cluster centers gave the best results.

1.3 Classification Discussion

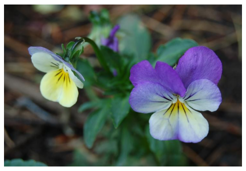
1.3.1 Performance Evaluation

Granted our combination of two non-deterministic methods, the highest average validation accuracy we received was 0.22 and the lowest 0.11. The clustering of the SIFT descriptors is a non-deterministic process and can result in largely different visual words. Our accuracy on the test set, and our submission to the Kaggle Challenge, was an unprecedentedly low 0.02, which we believe is due to incorrect transcribing of image numbers for the final csv.

1.3.2 Intricacies of Classification Methodology

The suspected sources of error caused by our chosen methodology come in two forms, the lack of distinction between foreground and background when using SIFT features, and the selection of cluster centers. The issue in the lack of distinction between foreground and background is that many of the visual words could have pertained to identifiable structures in the background of an image. This background element is useless, if not misleading for our classification. The next issue with the methodology is the selection of cluster centers, or equivalently, the number of visual words to be used. As our dataset contained 102 categories, given the pigeon hole principle, we should use at least 102 visual words. Much research has gone into determining the optimal number of cluster centers, but through experimentation we found much variance.

1.4 Classification Examples


True 52/Pred 34

True 52/ Pred 52

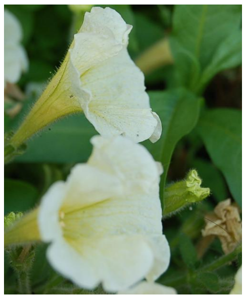
True 52/ Pred 52

True 52/ Pred 52

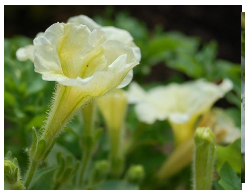
True 52/ Pred 52

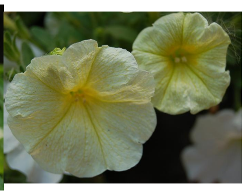
Category 52 Examples

True 51/Pred 51

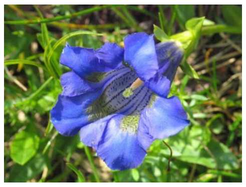

True 51/ Pred 79

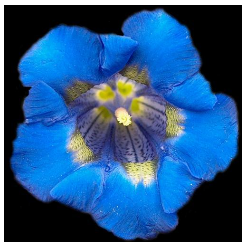
True 51/ Pred 51

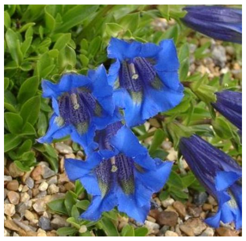

True 51/ Pred 7

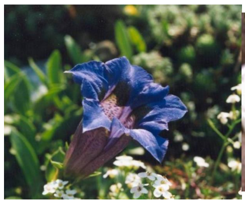

True 51/ Pred 38

Category 51 Examples


True 28/Pred 18


True 28/ Pred 102


True 28/ Pred 77


True 28/ Pred 83

True 28/ Pred 28

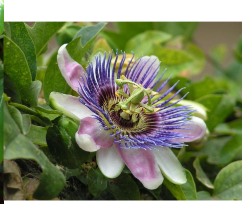
Category 28 Examples

True 77/Pred 77

True 77/ Pred 77

True 77/ Pred 77

True 77/ Pred 77


True 77/ Pred 77

Category 77 Examples

True 94/Pred 94

True 94/ Pred 46

True 94/ Pred 94

True 94/ Pred 97

True 94/ Pred 94

Category 94 Examples

2 Segmentation

In order to segment the flower images, we first filtered the images to reduce noise; we then then utilized a *Gaussian Mixture Model* (GMM) with *Expectation Maximization* (EM), and a *K-Means* (KM) model to perform segmentation. Implementation and hyperparameter details for these models are discussed in **Section 2.2**.

2.1 Segmentation Results

2.1.1 Confusion Matrix

Below are the confusion matrices for both segmentation methods. Each data point represents one pixel, with 'Yes' representing a white pixel in the segmentation mask (or the foreground/flower) and 'No' representing a black pixel (or background). Ground truths were provided in the dataset, while the predictions were generated by each segmentation method.

Gaussian Mixture Model:

		Ground Truth	
		Yes	**No**
Predicted	**Yes**	7513727	4853286
	No	4895786	16159935

K-Means:

		Ground Truth	
		Yes	**No**
Predicted	**Yes**	1003731	2319220
	No	2372182	18694001

2.1.2 DICE *(Overall)*

Segmentation Model	**Avg. DICE**	**Std. Dev. DICE**
GMM	0.5283	0.2952
KM	0.7470	0.3013

2.1.3 DICE *(Per Validation Set)*

Gaussian Mixture Model:

Validation Set	Avg. DICE	Std. Dev. DICE
1	0.4945	0.2461
2	0.3630	0.2254
3	0.6029	0.2855
4	0.7720	0.2906
5	0.4092	0.2254

Normalized Graph Cut:

Validation Set	Avg. DICE	Std. Dev. DICE
1	0.6973	0.2637
2	0.6441	0.3680
3	0.6793	0.3362
4	0.8852	0.2189
5	0.8225	0.2132

2.2 Segmentation Methodology

2.2.1 Image Preprocessing

The images provided in *flower_segmentation/images* were preprocessed using a filter to reduce noise. The filter was applied to smooth the flower shapes, as an inspection of the masks provided in *flower_segmentation/segmentation* showed that the ground truth masks were fairly smooth and not extremely precise.

We decided to use a *bilateral filter* to achieve smoothing, while still maintaining strong edges in the images. A bilateral filter is an extension of the Gaussian Smoothing filter seen in class, but the filter weights also take into account differences in pixel intensity and color in addition to the standard distance between pixels. This helps maintain edges between regions of starkly different colors. The filter, therefore, should maintain the delineation between a flower petal and its background.

2.2.2 Segmentation Approaches

The approaches used for segmentation were selected following an comparison of the input images to their expected masks. It appeared to us that the main input feature tying an image to its ground truth mask is the difference in color between a flower and the background (usually green from leaves or brown from dirt). Therefore, we decided to approach the segmentation using colour-based clustering.

Gaussian Mixture Model: Our initial approach was to use the *sklearn.mixture.GaussianMixture* implementation in a supervised approach. We realized, however, after hours of trying to train the model that our personal laptops had insufficient compute power to iterate this process for hyperparameter selection. Therefore we utilized helper functions written by group member Durham Abric in *Assignment 3* for our GMM implementation, and made changes to refactor for the binary segmentation required. This implementation was unsupervised, and thus hyperparamter tuning had to be accomplished manually.

K-Means Clustering: The implementation for KM was based on the *skimage.segmentation.slic* function, which provides k-means clustering in the colour-(x,y) space. We coded a function to wrap the *slic* implementation, and provide additional functionality.

Note: Both approaches were initially hampered by the randomness of which label (of {0,1}) was applied to the foreground (flower) and background. On account of this, some of the predicted masks were inverted with a white background and a white foreground. In order to overcome this, we calculated the average distance of each label from the center of the image and coloured the label nearest to the center white (assuming the flower is central to the image) and the other label black. This caused a significant improvement, but still fails in some cases.

2.2.3 Hyperparameter Selection

Gaussian Mixture Model: The GMM implementation we used has only one hyperparamater: the number of EM iterations. In order to find the optimal number of iterations, we tested various values on a logarithmic scale (i.e. 1, 10, 100, 1000) and noted the DICE values for all. It became clear that the algorithm performed best with few iterations, and became too receptive to noise (e.g. lighting conditions, small textures) with large numbers of iterations. Therefore we honed in on iterations in the range 1-10, and tested all values in this range. The best DICE score was accomplished with 3 iterations.

K-Means Clustering: The KM implementation and the *slic* function took 3 hyperparameters: compactness, enforcing connectivity, and the maximum iterations. The most impactful of these parameters was compactness, which controls the balance between clustering on colour proximity and space proximity, such that low values highly weight color and larger values put more weight on location. Similar to the number of iterations in GM, we varied *compactness* logarithmically (i.e. 0.01, 0.1, 1, 10, 100) and honed in on the best value (holding all other parameters constant), which was 1.5. The same process was then repeated for the *max_iters* parameter, and we selected 20 iterations. In all instances, KM performed best with connectivity was *not* enforced, even though most foreground segments are contiguous in the ground truth masks.

2.2.4 Cross Validation

Both methods were tested on the input set as a whole, as well as on 5 distinct subsets on inputs that together comprised the entire input set. Because both segmentation approaches were unsupervised, it was unnecessary to segment the input into *training*, *validation* and *test* sets, because there isn't any feedback into the algorithm in order to improve performance. Thus, testing the algorithms on separate sets alone should quantify how well they will generalize to other examples of the same problem. The results of the cross validation approach can be seen previously in **Section 2.1.3**.

2.3 Segmentation Discussion

2.3.1 Performance Evaluation

As clearly outlined in **Section 2.1**, the K-Means Clustering approach was more successful than the Gaussian Mixture Model approach. Overall, we believe the results of the segmentation are strong, given the intersection between the ground truth and predictions are between 50-75% for both methods. However, both methods also have fairly high standard deviation for their respective DICE scores; analysis of why this is the case will be provided for each approach below in **Section 2.3.2**.

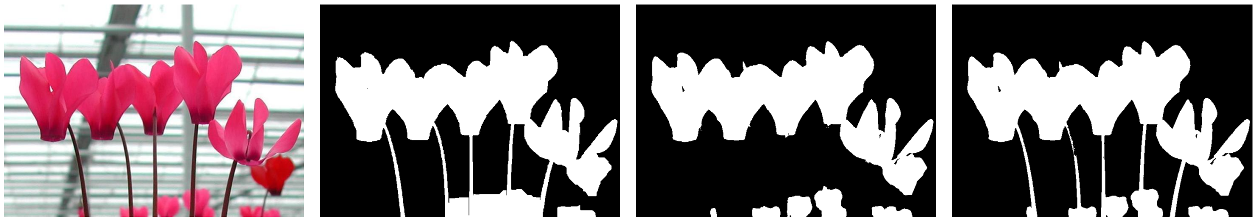
For both approaches, we believe that an implementation that automates hyperparameter tuning on a *per image* basis rather than on the dataset level. For both approaches, the ideal hyperparameters vary widely for different images; the contrast in colour is the main factor that varies which hyperparameters should be chosen.

2.3.2 Methodology Strengths & Weaknesses

Gaussian Mixture Model: The GMM model performs very well on flowers that have little variance in their color (see images *524* and *685* in **Section 2.4**), but often fails to identify the entire flower for multi-coloured flowers (see images *902* and *269* in **Section 2.4**). This may be because for our implementation clustering is entirely based on colour and not any spacial data. Therefore the variance in flower colours introduced a lot of variance in performance & therefore the standard deviation in its DICE score. Additionally, because the number of EM iterations was constant, the algorithm didn't always converge; as a result some predicted masks were coloured almost entirely white, which introduced many *false positive* pixels into the confusion matrix.

K-Means Clustering: In contrast to the GMM model, the KM implementation *did* use spacial data in its clustering, which is why it achieved a much stronger performance. Because spatial data was taken into account, the performance of KM was much more robust to colour variance within a flower. While the GMM model was prone to *false positive* labelling, the KM model was prone to producing entirely black predictions full of *false negative* results. Again, this may be because *max_iters* was set to 20, and some images may have required additional iterations to achieve the optimal result.

2.4 Segmentation Examples



Input Image Ground Truth GMM Prediction KM Prediction

Image 00524 Segmentation

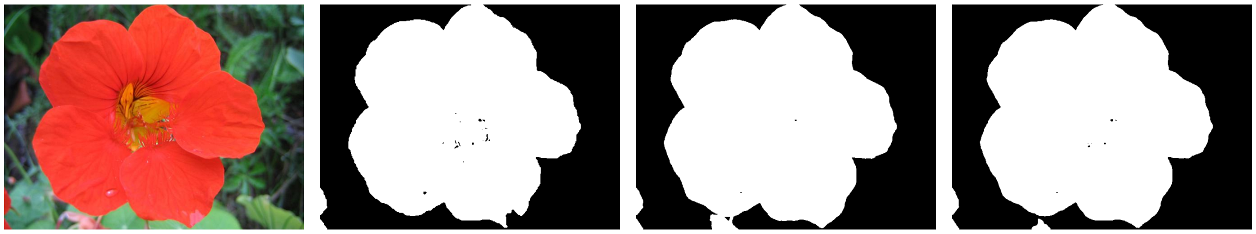

Input Image Ground Truth GMM Prediction KM Prediction

Image 00685 Segmentation

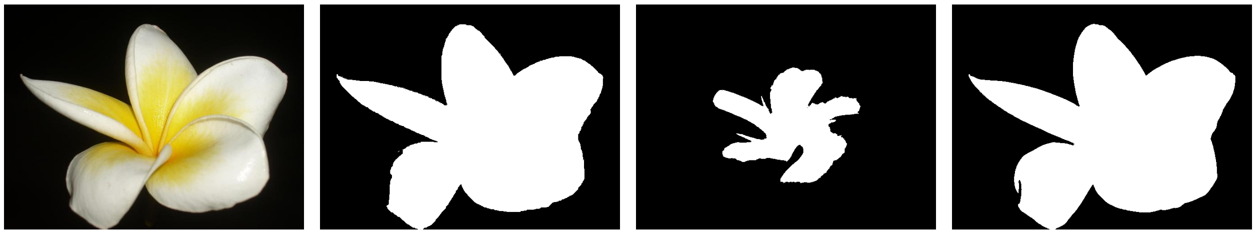

Input Image Ground Truth GMM Prediction KM Prediction

Image 00902 Segmentation

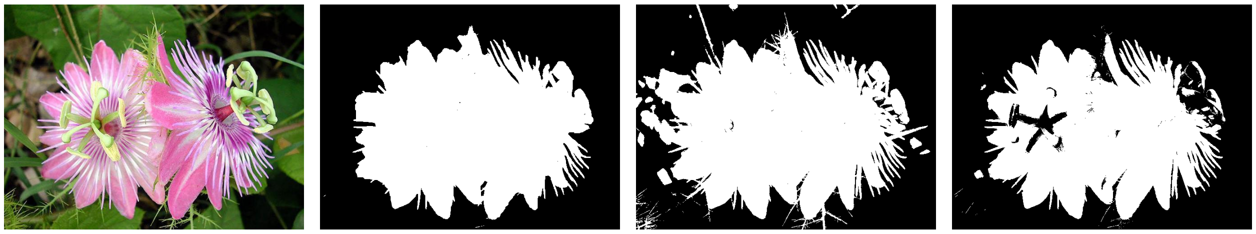

Input Image Ground Truth GMM Prediction KM Prediction

Image 00083 Segmentation

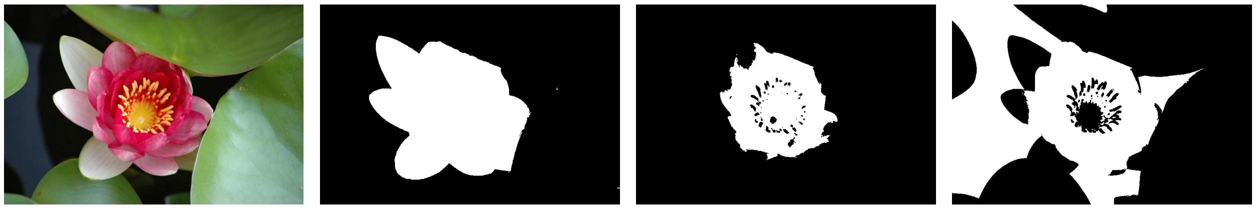

Input Image Ground Truth GMM Prediction KM Prediction

Image 00269 Segmentation