Computer Vision Coursework 1 - Randomised Decision Forests

Paul Streli - ps4715
Imperial College London
CID: 01103106

Karoly Horvath - kth15
Imperial College London
CID: 01088730

Abstract

The following paper examines different machine learning algorithms for object categorisation, using the Caltech101 dataset. A focus is put on the optimisation of the Random Forest (RF) Classifier and the influence of Bag of Words codebook generation using K-means clustering and Random Forests on classification accuracy. The best performance was achieved with a K-mean codebook of size 200 and a RF-Classifier with 30 trees of depth 8.

1. Introduction

This paper is concerned with visual object categorisation, the problem of determining what object is present on a particular picture. Several visual object classification methods were tested on how well they overcome the challenges of large intra-class variation. The experiments are performed on 10 object categories (classes) from the Caltech101 dataset. [2] 15 training and 15 testing images of each class were randomly selected, resulting in a total of 150 training and 150 testing images.

2. K-Means Codebook Generation

First, a feature vector needs to be extracted from each image that can be used as input for the classifier. This is done using the Visual Bag of Words method, which is similar to the classic text-based Bag of Words approach (described in Appendix 6.1).

2.1. Vector Quantisation Process

The extracted SIFT descriptors, which were proved to achieve good results for image classification problems (described in Appendix 6.2), provide the input for the codebook generation process [3]. Following the method described in Appendix 6.3, the descriptor ensemble is clustered using the K-means algorithm (k = *size of the codebook*), and the cluster centres are selected as the *codewords*. The feature vectors for the training and testing images are built by counting how many of their descriptors are closest (in the Euclidean sense) to each codeword (cluster centre), in the 128 dimensional space. These feature vectors are used for classification (one feature vector per image - in the dimension of the codebook size). The left side of Figure 1 shows three training images and the right side shows three testing images of a particular class (water lily) with their corresponding histograms (feature vectors), generated with the K-means clustering method using *k=15* (for visualisation). It can be seen from the first two training and testing images that the histograms do bear resemblance, which is desired for good classification. However it is important to notice that as the third row shows, some histograms are not that close to the others, which might lead to deteriorating classification accuracy. Further histograms and analysis can be found in Appendix 6.4.

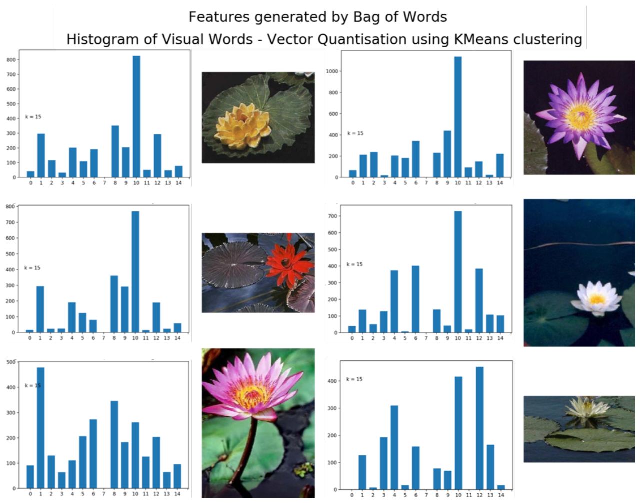


Figure 1: Histograms for the *Water Lily* class

2.2. Evaluation

As can be seen from Fig. 7 (see Appendix 6.5), the time for clustering increases linearly with codebook size as expected from the theoretical complexity seen in lectures ($O(DN'k)$). Interestingly, the memory consumed by the clustering process seems to remain constant from a certain point on. A more detailed analysis of the influence of the codebook size on the distribution of the image histograms is conducted in Appendix 6.5.

3. Random Forest Classification

After obtaining the image histograms, it is necessary to find a suitable method for object categorisation. As a baseline classification method, Random Forest Classification (see Appendix 6.6) was performed using the unnormalised K-Means histograms as input. The quoted results are obtained using the test set for validation (see Appendix 6.7).

Moreover, as the Random Forests are randomly generated, their prediction accuracy can vary even if the parameter settings and datasets are unchanged. This variation can be quite significant, especially for forest ensembles with a low number of trees. Therefore, the classification process was repeated five times for each setting, and the mean classification accuracy and its standard deviation were calculated.

Using the measure of *entropy* as objective function and the axis-aligned split function, the hyperparameter optimisation was thoroughly done over the number of clusters k for K-means, T, D, s and m (see Appendix 6.6). The best mean classification accuracy obtained was 0.701 (See Table 1). Changing the weak learner function to two-pixel did lead to a slight (within standard deviation) improvement in classification accuracy, but added computational time and complexity. Using *gini gain* as the objective function increased the classification accuracy to 0.755, which is the best result so far. From the confusion matrix generated on the unseen test data, it can be observed that the classifier performs very well on the classes with distinct shape and mainly unchanged observer angle (birds-eye perspective on the Trilobite class and the Yin-Yang) but seems to have difficulties learning the underlying distribution when the form and the viewing points differ within the class (in the case of Watches and Umbrellas) (see Appendix 6.8 for class index map and Appendix 6.9 for example images). These problems could be eased by collecting a larger dataset.

When examining the hyperparameters D and T, it was noted that classification accuracy increased with both, but the increase flattened out after a certain point. For T, a peak around 20 decision trees was found (see Figure 17a, App. 6.14).

The maximum tree depth is bounded by the number of training samples and the minimum number of samples in a leaf node. If $m = 2$ and there are 150 training samples, the tree cannot grow deeper than a depth of 10 ($2^{10-2} > 150$), however the plateau in classification accuracy occurs even earlier at a depth of around 5. The other parameters play only a minor role as long as they are chosen within a good region. The number of dimensions that are considered for each split, controlling the randomness parameter, should be bigger than 20 for a codebook size of 200 (see Figure 17c, App. 6.14) and the minimum number of samples per leaf node may be between 2 and 7 without negatively affecting

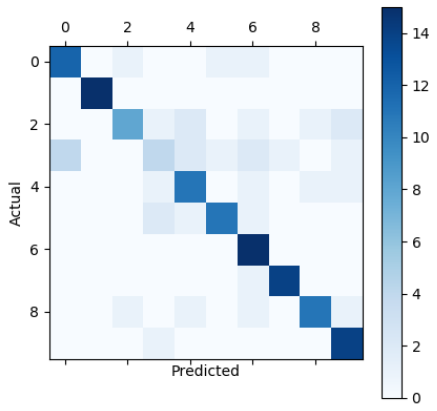


Figure 2: Confusion matrix of best Random Forest Classifier with 200 K-Means clusters

RF Class.	k	T	D	s	m	obj.	weak	Acc.	Std.
base A	50	20	8	50	2	ent.	axis	0.701	0.013
base B	200	20	8	50	7	ent.	twop.	0.726	0.019
base C	200	30	8	20	5	gini	axis	0.755	0.013
norm.	200	30	8	20	5	gini	axis	0.753	0.008
PCA-50	200	30	8	20	5	gini	axis	0.607	0.014

Table 1: Classification accuracy for different RF Classifier variants (optimal parameter settings and mean test accuracy for five iterations are shown)

accuracy (see Figure 17d, App. 6.14). The influence of bagging and the bag size for each tree, as long as not taken to extremes, did not have a significant impact on the recognition accuracy obtained.

Increasing the codebook size (number of clusters in K-means), has a positive impact on recognition accuracy (see Fig. 9, App. 6.10). There is a strong increase in accuracy for a rise in the codebook size for small codebooks. Then, the function flattens out, reaching a peak around $k = 200$. Given this and the fact that clustering time increases significantly with codebook size, k should be around 200. Smaller feature dimensionality also improves the generalisation properties of the classifier. Figure 3 shows how training and testing time varies with codebook size. Testing time remains almost constant with codebook size and is negligible compared to testing time, highlighting Random Forest classification's suitability for real-time applications. Memory and training time increase with codebook size but start to flatten out. The effect of the codebook size gets smaller, as only a fixed amount of dimensions are randomly checked in each split. Further improvement suggestions can be found in Appendix 6.11.

To compare these results to an alternative classification algorithm, a Support Vector Machine classifier (SVM) was

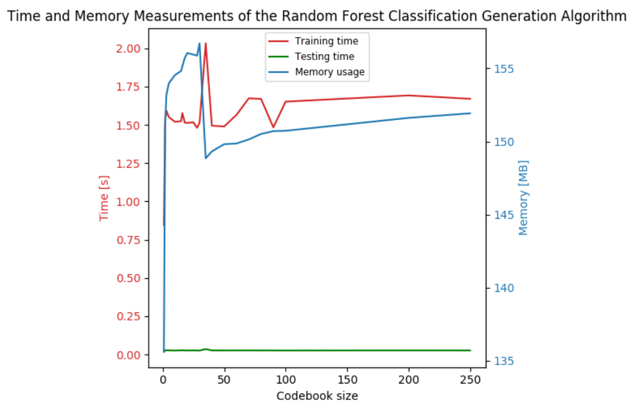


Figure 3: Training and testing time for different codebook sizes, (K-Means histograms, $m = 2$, $s = 5$, $T = 20$, $D = 10$, weak learner: axis, objective: entropy)

implemented that determines the class label based on the K-mean histograms. As the results in Appendix 6.12 show, Random Forest Classification does perform significantly better.

4. Random Forest Codebook Generation

RFs provide a supervised alternative for codebook generation. For this, decision trees are grown as described in Appendix 6.6 but instead of using the image histograms as inputs, the raw descriptors are clustered by the trees according to the specified objective function and the image class they were extracted from. Afterwards, the leaf nodes are ascendingly indexed across all trees so that each leaf node of a tree corresponds to a single codeword. Finally, each image's descriptors are pushed through the trees and the count for the codeword is increased by one if a descriptor ends up in the corresponding leaf. In this way, each descriptor adds a number of hits equal to the number of trees in the ensemble to the histogram.

To assess the vector quantisation complexity, the time and memory usage of the algorithm was measured. Figure 12 in Appendix 6.13 shows that the computation time grows again linearly with codebook size. Moreover, generating the codebook using Random Forests was 33% slower than using K-Means clustering. Both is unexpected from the theory presented in lectures. This behaviour might be caused by the unoptimised nature of the implementation.

The hyperparameters for the RFs responsible for the codebook generation and the settings for the classifying RFs were optimised over a wide value range. While it was expected that the classification accuracy would increase due to the supervised nature of the codebook generation process, the highest mean accuracy with optimal settings was only measured to be 0.600 (see Table 2) and thus clearly lower

Class.	Classifier Parameters	Acc.	Std.
RF.	$T = 20$, $D = 8$, $s = 50$, $m = 5$, ent., axis	0.600	0.014
RF norm	$T = 20$, $D = 8$, $s = 50$, $m = 5$, ent., axis	0.599	0.021
RF PCA 50	$T = 20$, $D = 8$, $s = 50$, $m = 5$, ent., axis	0.453	0.014
SVM	$\gamma = 0.001$	0.110	-

Table 2: Best mean accuracies for different classifiers (optimal RF Codebook settings: $T = 10$, $D = 5$, $s = 5$, $m = 2$, objective function: entropy, weak learner axis; mean test accuracy and standard deviation are shown for five iterations)

Class.	Codebook	Parameters	Best accuracy
RF-Class.	K-means	base A: see Table 1	0.755
RF-Class.	RF-Codeb.	RF. base: see Table 2	0.600
SVM	K-means	$k = 100, \gamma = 10^{-3}$	0.440

Table 3: Summary of final results

than in Section 3. A possible reason could be the small amount of images per class paired with the high amount of descriptors per image. Possibly, the intra-class descriptor variance was too high for the RFs to learn the underlying distribution causing the RFs to overemphasis certain leaf nodes (see Fig. 15 and 16, App. 6.13). From Fig. 13 (Appendix 6.13), it can be observed how the wrong predictions have spread out more evenly across a wider range of classes compared to Fig. 2. This can be only somewhat counteracted by increasing the codebook size via an increase in depth and number of trees. As can be seen from Fig. 14a and 14b, the accuracy rises only very slowly from a certain point on. For smaller codebook sizes, it is better to have a higher number of trees. In higher dimensions, it seems to be of less importance whether deeper or a higher amount of trees are grown.

A SVM Classifier performed very badly on the new histogram set achieving an accuracy of only 0.110 which indicates that the SVM failed to learn anything meaningful (all images are predicted as class 9) and underlines once more the strength of RFs for classification. Again, normalisation did not improve accuracy and dimension reduction using PCA led to a decrease (see Table 2).

5. Conclusion

Overall, the experiments have shown the strength of the RF classifier over alternative classifier algorithms for image categorisation (see Table 3). The best mean accuracy was obtained using 200 clusters for K-means and a RF classifier with 30 trees and a depth of 8. Unexpectedly, K-means codebooks were superior to their supervised alternative using RFs in achieving higher classification accuracy. For future work it would be interesting to see if this trend prevails for larger datasets.

6. Appendices

6.1. Appendix A - The Classical Bag of Words Method

It is a common problem to classify various form of texts, like emails, books, text messages, etc. For example spam filters can classify incoming emails as spam or 'ham' (not spam). The Bag of Words feature extraction method has been proved to work well to generate the features for classification, originally using textual data, where it has got its name from. The textual dataset consists of words, which are called *codewords*, and the ensemble of the codewords is called a *codebook*. The feature vectors for each text instances are constructed from the number of times each codeword appears in the text (this might be zero). Once every text's feature vectors has been built this way, the classification problem can be tackled. The feature vectors from the test instances are constructed following this method and are used to decide which category they belong to. More sophisticated variations of this approach exist, one of which groups N consecutive words (called N-grams) to create the codebook. This allows the model to capture a bit more meaning from the documents.

However, while in the original Bag of Words method, it is compelling to build the feature vectors ("referred to as histograms") by counting how many words or (N-grams) are found in a text (called *codewords*), for the images first it is necessary to establish the visual *codewords* used to build the histogram, and the method of counting how many times they appear on the sample picture.

6.2. Appendix B - Obtaining the Image Descriptors

It is necessary to extract appropriate dense SIFT descriptors from all of the testing and training images that are repeatable, general enough to appear in multiple images and invariant to translation, reflection, rotation and geometric and illumination variations, following the method. Given 30 images from 10 classes, first appropriate descriptors from the pictures are obtained using the VL_Feat library [5], whose VL_PHOW function applies dense Scale-invariant feature transformation (SIFT) to each image [6]. The grid centers used for the algorithm are defined as every 8^{th} pixel of the image in this case. The SIFT descriptors are computed on the 4x4 neighbouring patches of the grid centers by calculating the Gaussian derivatives over 8 orientation planes, which results in a descriptor size of 128. In the following experiments, the image samples were converted to grey scale and downsampled with a rate of 4, 8 and 10, and the descriptors were calculated on these transformed images. Using these descriptors ensures that the codewords used for feature generation are repeatable, general enough to appear in multiple images and are invariant to translation, reflection, rotation and geometric and illumination variations.

Note that since the images are not of the same size, different amount of grid centers will be contained in every picture. This leads to different amount of descriptors being extracted for each image.

6.3. Appendix C - K-Means Codebook Generation

After obtaining the large amount of descriptors (data points) for each image in the 128 dimensional space, it is necessary to select the most useful codewords. To achieve this, all of the descriptor data points of all of the images are placed into a 128 dimensional space and they are clustered into *k* clusters using the K-means clustering algorithm. The codewords are taken as the cluster centres, therefore the vocabulary size is given by *k*. In the visual Bag of Words case the descriptors of the training and testing images are compared with the cluster centres using the nearest neighbour method. Each testing and training image is placed in the space containing all of the cluster centres. Using nearest neighbour method in the 128 dimensional space, the closest codeword (cluster centre) to all of the descriptors is determined, using Euclidean distances. The classification feature vectors are produced from the count of how many descriptors were matched to each cluster centre. They are *k* (vocabulary size) dimensional, which is therefore the input dimensionality of the classification problem.

6.4. Appendix D - Evaluation of Features Generated by K-Means

On the left side of Figure 4, training images and on the right side testing images of the Yin Yang class are shown, with their corresponding feature vector histograms generated with a codebook size of *k=15*. Similarly to the observations made on Figure 1, the histograms in the first two rows are similar, which demonstrates that the generated features are suitable for classification. It is important to note that the histograms of this class look distinctively different than the histograms generated for the *Water Lily* class, which leads to the belief that the classifier will be able to distinguish between different classes. However, again, the third row displays a histogram that is of a different shape, which could cause the classifier to have difficulties in selecting the correct class label for the given image.

Figure 5 shows a full-sized histogram (feature vector) that was created with a codebook size of 200.

6.5. Appendix E - Analysis of the K-Means Clusters Created

Alongside the traditional measures of the training time and memory usage, two custom measures were defined to evaluate the influence of the codebook size, which are based on the similarity of the histograms that belong to the same class of images and the dissimilarity of those that do not.

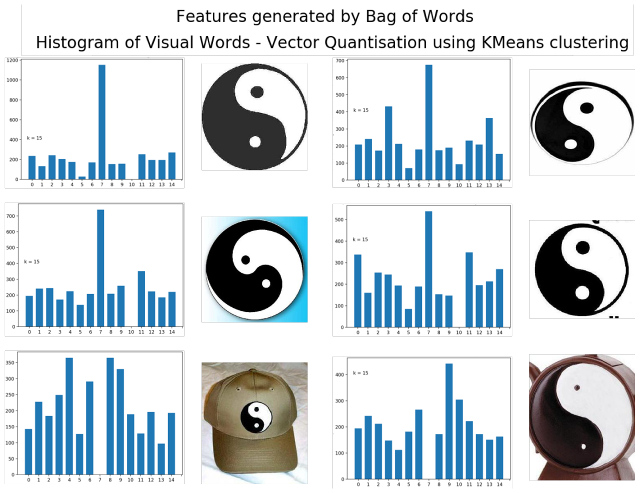


Figure 4: Histograms for the *Yin Yang* class

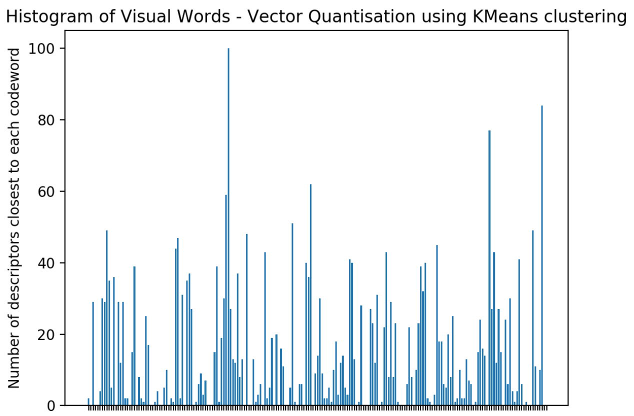


Figure 5: Histogram for a *Water Lily* image using *k=200*

The generated feature vectors should be close to each other if they belong in the same class, while far away from each other if they belong to different classes. The nearest neighbour mean average precision value of the feature space on 20 nearest neighbours and the ratio of average intra-class vs. inter-class Wasserstein distances of samples are graphed against the vocabulary size *k*, as shown on Figure 6. Based on these, and the time-memory graph presented on Figure 7, it is established that the best classification accuracy within time and memory constraints is expected to be achieved using the feature set generated by a codebook of size 50. However, these measures do not necessarily need to correlate with the classification accuracy achieved. The influence of codebook size on accuracy needs to be further evaluated in Section 3. As it was revealed later by the Random Forest Codebook Generation, the RF codebooks yielded a worse categorisation accuracy (Fig. 11, App. M) compared to their K-means alternative even though they achieved a better Wasserstein score on average. mAP score was indeed better for K-means and could thus be potentially be a better metric for the quality of the codebooks. However, this would need to be checked on other datasets as well.

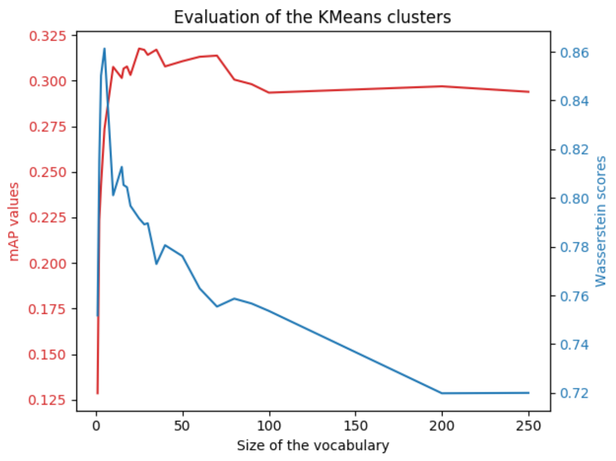


Figure 6: mAP and Wasserstein values of the features using K-Means Codebook Generation

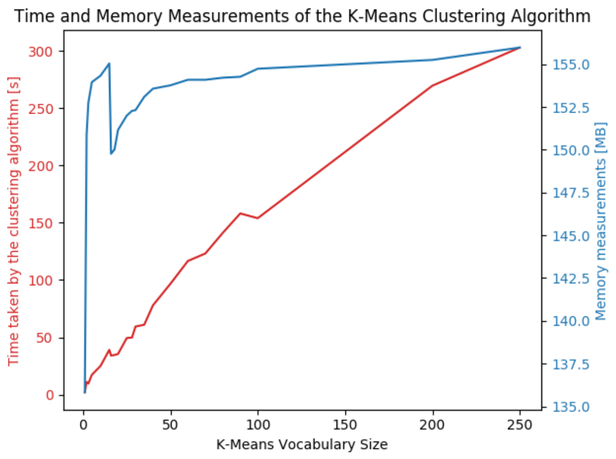


Figure 7: Time and memory measurements of the K-Means Codebook Generation

6.6. Appendix F - Random Forest Classification Method Overview

Random Forests (RFs), also known as Decision Forests (DFs), have gained in popularity for their applications in Computer Vision in recent years. An RF is an ensemble of randomly different decision trees that ideally lead to better generalisation properties and higher robustness through their decorrelation via randomised feature and training set sampling. Due to their architecture that also facilitates parallel execution on a GPU, decision trees work well for multi-class problems, can handle high-dimensional input data efficiently and can scale up to large datasets.

During training, each decision tree receives a randomly bagged (i.e. sampling with replacement) subset of the training set as input. The tree itself consists of nodes that contain a split function, also referred to as weak learner, that determines whether a certain sample is passed on to either the left or right child node. This split function can be linear or even non-linear over several input dimensions but usually due to speed advantages and simplicity does a single comparison on a selected input feature dimension. The latter is known as axis-aligned weak learner and can be represented as $h(\mathbf{v}, \theta) = [x_i < \tau]$ for any $i \in \{1, ..., d\}$. Here, $h(\mathbf{v}, \theta)$ is the split function, $\mathbf{v}$ is the input sample ($\mathbf{v} = (x_1, ..., x_d) \in \mathbb{R}^d$), τ is the split threshold and θ symbolises the node test parameters such as threshold, which feature dimensions to consider and how to combine them. Alternatively, the two-pixel test can be used which compares the difference between the values of two different dimensions of a given sample, closely resampling a discrete differentiation, in the form of $h(\mathbf{v}, \theta) = [x_i - x_j < \tau]$ for any $i, j \in \{1, ..., d\}$ and $i \neq j$.

The randomised feature selection for each split node can be implemented via randomly selecting a dimension and a threshold between the subset's minimum and maximum value. This can be done for several iterations and the settings that maximise a given objective function can be taken as final selection. The randomness factor decreases with the amount of random trials as it is more likely that the different tree's obtain similar local or global maxima. As objective function, especially for discrete probability distributions, the information gain can be used. It is defined as follows:

$$I_E(S, \theta) = H(S) - \sum_{i \in \{L,R\}} \frac{|S^i|}{|S|} H(S^i) \tag{1}$$

$I_E(S, \theta)$ is the information gain for the the node with parameter θ and set S of samples reaching that node. $|S^i|$ represents the number of samples in that are put into the left or right child node and $H(S)$ is the Shannon entropy defined as:

$$H(S) = -\sum_{c \in C} p(c) \log(p(c)) \tag{2}$$

c represents a class label and is part of the set of all class labels C. $p(c)$ is the empirical distribution of that class label within the node data set S. The entropy increases as the distribution spreads out more uniformly across all classes.

Otherwise, it is also possible to use the decrease in *gini impurity* [1] (i.e. gini gain) as objective function which is a measure of the chance that a randomly chosen sample from the node subset is incorrectly classified when the class label is randomly chosen according to the node distribution. Mathematically, the objective function is defined as follows ($G(S)$ is the Gini impurity):

$$I_G(S, \theta) = G(S) - \sum_{i \in \{L,R\}} \frac{|S^i|}{|S|} G(S^i) \tag{3}$$

$$G(S) = 1 - \sum_{c \in C} p(c)^2 \tag{4}$$

Eventually, as splitting continues and the tree grows in depth, there will need to be a condition to stop. The final nodes are called leaf nodes and they are likely to contain a more class-selective subset of the original training dataset. The stopping criteria can be defined as maximum tree depth, minimum node subset size or the case of not being able to find a split that leads to an information gain above a certain threshold. Depending on the exact implementation, the policy might lead to asymmetric and not fully grown trees that facilitate higher decorrelation and diversity across the ensemble and therefore might have positive generalising effects.

For classification, a test sample is put through all trees. Each time, it takes the path defined by the split function until it ends up in a leaf node. Each leaf node incorporates a discrete probability distribution prescribed by the class occurrences in the corresponding training leaf subset. As samples of the same class are likely to end up in the same leaf nodes due to the shared similarities in the feature vectors, this distribution can be assigned to the corresponding sample. The test histogram will receive a distribution from each tree that can be merged via averaging the different posterior distributions. Finally, the class with the highest probability can be assigned to the test sample.

Overall, there are the following hyperparameters that can be optimised for a given task:

- **Number of trees:** T
- **Maximum depth of trees:** D
- **Random trials per split to optimise over:** s
- **Minimum samples within a node to stop splitting:** m
- **Fraction of original input size for bagged tree training data**
- **Tree objective function: entropy vs. gini**
- **Split function type: axis vs. two-pixel**

Due to the lack of a deeper theoretical framework, the search for the optimal settings usually needs to be done on a trial and error basis.

6.7. Appendix G - Validation

The training set proved to be too small to yield meaningful results when it was divided into training and validation subsets. Using a set size of five training images per class as validation set, the suggested optimal hyperparameters varied and did not closely yield the best results on the test set due to the small amounts of images per class. Therefore the larger test set was decided to be used for validation when future results are quoted. While the test error loses its validity as completely unbiased estimator of the true classification accuracy on unseen test data, it reduces the variance in finding the optimal settings and therefore is a justifiable approach for very small datasets.

6.8. Appendix H - Object Classes

- Class 0: Tick
- Class 1: Trilobite
- Class 2: Umbrella
- Class 3: Watch
- Class 4: Water Lily
- Class 5: Wheelchair
- Class 6: Wild Cat
- Class 7: Windsor Chair
- Class 8: Wrench
- Class 9: Yin Yang

6.9. Appendix I - Success and Failure Cases

While the left picture was correctly classified as a water lily, the classifier failed to correctly categorise the right picture. It was labelled as a wrench. This cannot be really understood by the human observer but could be due to the change in observer angle and the fragility of the constructed RF classifier.

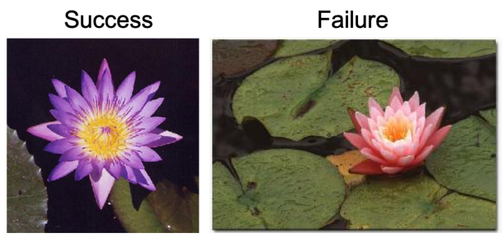


Figure 8: Confusion matrix of best Random Forest Classifier with 200 K-Means clusters

6.10. Appendix J - Recognition Accuracy vs. Codebook Size

6.11. Appendix K - Improvements on the Random Forest Classifier

Another possible improvement was to normalise the histograms so that each histogram would represent a probability distribution that sums up to 1 (i.e. divide by the total image's descriptor number). However, this did not really change recognition accuracy for the Random Forest Classifier with optimal settings (see Table 1).

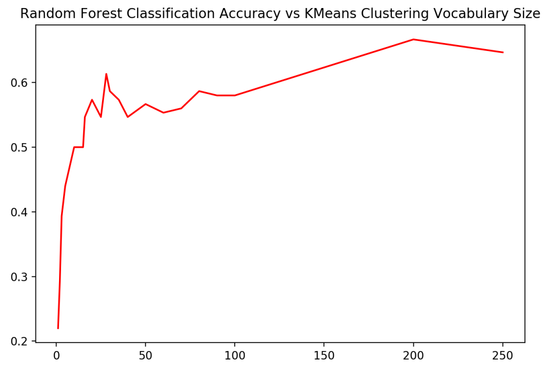


Figure 9: Number of clusters k vs. classification accuracy, (K-Means histograms, $m = 2$, $s = 5$, $T = 20$, $D = 10$, weak learner: axis, objective: entropy)

Since lots of the histogram bins had counts equal to zero, PCA dimensionality reduction was tested. For this, a histogram with codebook size of 200 was reduced to 50, 100 and 150 dimensions. However, this lead to a reduction in recognition accuracy in all cases (0.606, 0.580, 0.519 respectively).

6.12. Appendix L - Support Vector Machine Classification

To compare the results to an alternative classification algorithm, an SVM (Support Vector Machine) was implemented that determines the class label based on the K-mean histograms. For this, the *sklearn.svm.SVC* [4] class was utilised. The classifier was optimised over several values of k and different γ (logarithmic range $[10^{-3}, 10^{3}]$). As kernel type the Radial basis function was specified. The best classification accuracy on the test dataset was 0.440, obtained with 100 clusters and a gamma value of 0.001. The codebook size had a much bigger impact on the recognition accuracy than the exact value of γ. Normalisation of the histograms led to a reduction in accuracy (0.387). So did a change in the kernel function to polynomial (0.233) and dimensionality reduction with PCA to 50 dimensions (0.253). These result show that while SVMs are known to operate effectively on small and high-dimensional datasets, the RF Classifier and its architecture seem to be superior for the given image object recognition task, further underlining the current trend in popularity.

From the confusion matrix, it can be seen that the SVM Classifier tended to overpredict the Yin Yang class. This points to overfitting and is an indication that the SVM failed to learn the underlying distributions for most classes.

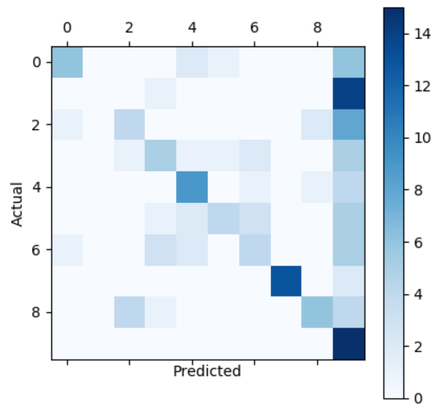


Figure 10: Confusion matrix of SVM Classifier with 100 clusters K-mean

6.13. Appendix M - Random Forest Codebook Figures

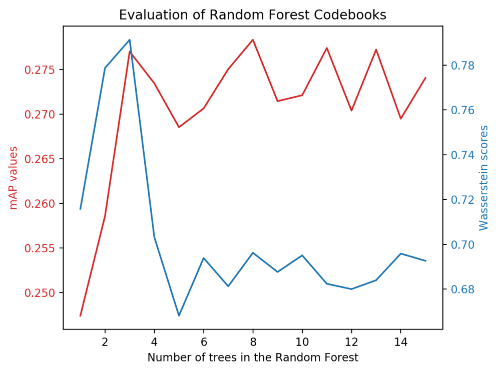


Figure 11: Evaluation of RF-Codebook generation

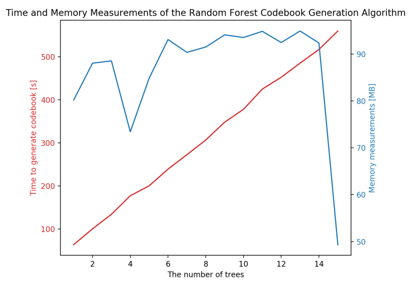


Figure 12: Time and memory for generating RF-Codebooks and the corresponding histograms

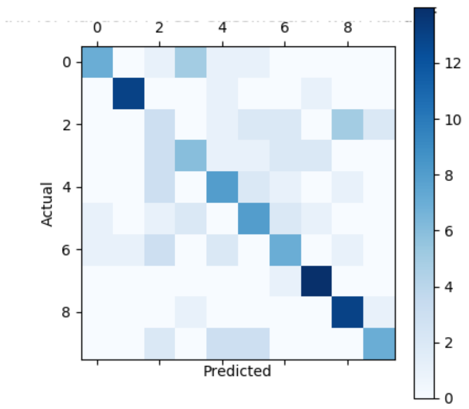
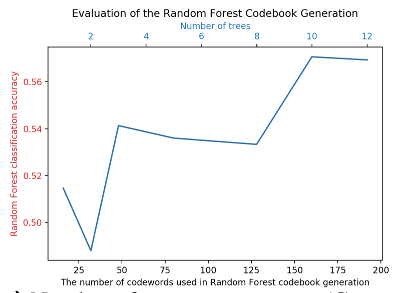
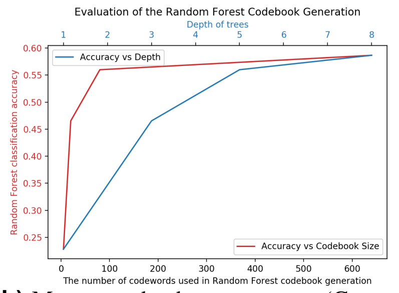


Figure 13: Confusion matrix of best RF Classifier with optimal RF-Codebook



a) Number of trees vs. accuracy (Generator: $D = 5$, $s = 7$, $m = 2$, entropy, axis)



b) Max. tree depth vs. accuracy (Generator: $T = 5$, $s = 7$, $m = 2$, entropy, axis)

Figure 14: Change in RF Classifier accuracy with changes of RF Codebook generator settings (Classifier: $T = 20$, $D = 8$, $s = 5$, $m = 2$, entropy, axis)

References

[1] B. Ambielli. Gini impurity (with examples). `https://bambielli.com/til/2017-10-29-gini-impurity/#`, 2017.

[2] F.-F. Li, M. Andreetto, and M. A. Ranzato. Caltech101 image

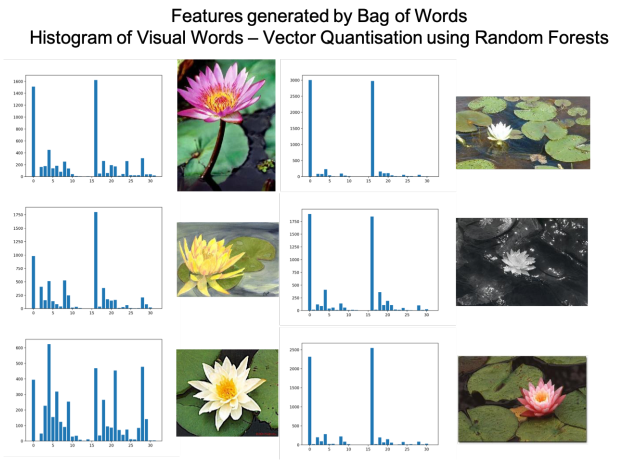


Figure 15: Vector Quantisation using Random Forest for Water Lilies

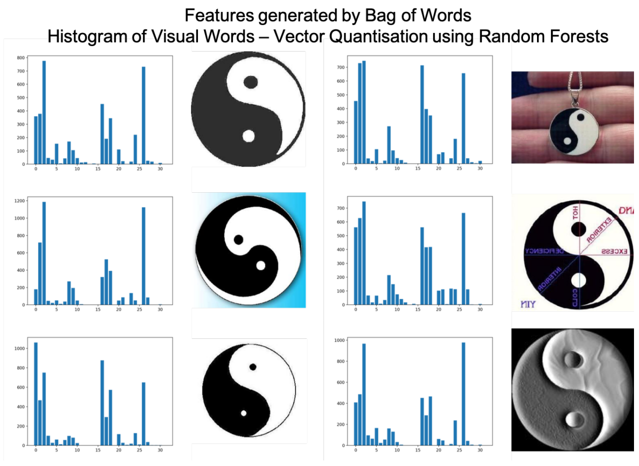


Figure 16: Vector Quantisation using Random Forest for Yin Yangs

dataset. 2003.

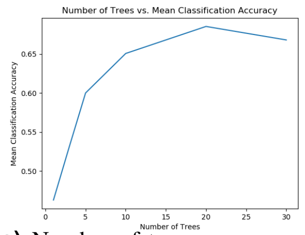
[3] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

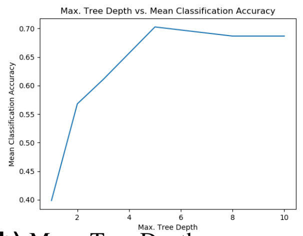
[5] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[6] Z. Zivkovic and B. Krose. On matching interest regions using local descriptors - can an information theoretic approach help? 01 2005.

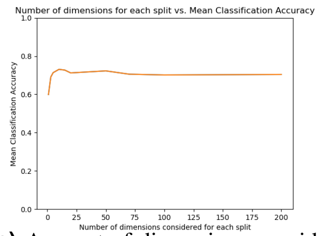
6.14. Appendix N - Evaluation for Different Tree Sizes



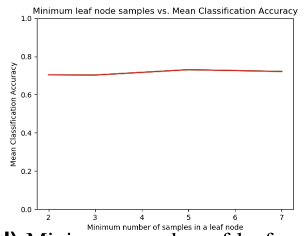
a) Number of trees vs. accuracy ($D = 8, k = 50, s = 50, m = 2$)



b) Max. Tree Depth vs. accuracy ($T = 20, k = 50, s = 50, m = 2$)



c) Amount of dimensions considered for each split vs. accuracy ($T = 20, D = 8, k = 200, m = 2$)



d) Minimum number of leaf node samples vs. accuracy ($T = 20, D = 8, k = 200, s = 50$)

Figure 17: Change in RF Classifier accuracy with certain hyper-parameters (K-means histograms, weak learner: axis, objective: entropy)