Lab 08 - PHOW

Jhony A. Mejia
Universidad de Los Andes
Biomedical Engineering Department
ja.mejia12@uniandes.edu.co

Abstract

Classification problems in computer vision have been one of the most interesting issues to be studied. The idea is to find a way to know that a certain image is part of a bigger category. The main principle is to find patterns between things in the same category and find differences between things from different groups. The method used in this paper for that objective is PHOW (Pyramid Histogram of Visual Words). The optimal PHOW's hyper-parameters were found for Caltech101 database and ImageNet200 database. The methods for training and evaluating were different for each database, and the optimal hyper-parameters were different as well. The results obtained were better for Caltech101 than the ones obtained in ImageNet200. To sum up, the performance of a method highly depends on the number of classes and the variation between each class. Better strategies than PHOW can be developed to study classification problems.

1. Introduction

PHOW (Pyramid Histogram of Visual Words) is part of the bag-of words (BoW) model, in which image features are treated as words. The main idea was originally used for document classification in text understanding, and was later applied to image recognition in computer vision. A bag of words (image features) represents the frequency of the words (image features) repeated in a text document (image). The main problem with the bag of visual words model is that the spatial information of image features are no longer available in the model representation. In BoW we know that a particular feature exist in the image, and we know how frequently, but we cant say where in the image. Pyramid histogram of visual words has been suggested to address this problem. This approach works by dividing the image into increasingly fine sub-regions, which are called pyramids. The histogram of visual words is then computed in each local sub-region [1].

PHOW features are based on dense SIFT descriptors. SIFT descriptors can be described broadly as descriptors of form-shape. Those SIFT features can be calculated for each of the training images and then be quantized using k-means clustering to form a visual vocabulary. After that, a spatial pyramid (of at least three levels) is created. The histogram of SIFT visual words that is calculated for each bin forms PHOW features. Despite the simplicity of the model it has shown promising results on large-scale datasets [1].

PHOW takes into account (indirectly) the spatial information, whereas SIFT does not. SIFT is a process for obtaining local information, whereas PHOW is usually used for obtaining more global information. It is important to say that PHOW is based on SIFT. In the basis, PHOW is scale invariant (because its basis is SIFT). However, after the sub-sampling process PHOW is no longer scale invariant. PHOW results depend strongly in the scale at which the histograms are being calculated.

2. Materials and methods

2.1. Datasets

Caltech101 is a Database developed by the California Institute of Technology for studying classification problems. The Database counts with 101 categories (actually 102 counting 'Faces_easy'). The categories are not balanced, which means that each category has different number of images. In general there are approximately 40-800 images per category. The size of each image is roughly 300 x 200 pixels. However there are a lot of images that exceed those dimensions. Each image is in RGB and its format is .jpg. Finally, the database is not divided in train and test categories.

ImageNet is a Database that also is focused on studying classification problems. The database has 200 categories, with 100 images per category (categories are already balanced). All the images are in format .JPEG and their resolution is 256x256. The database is divided in train and tests sets, each set contains 200 categories and 100 images. Taking this into account, the database actually has 200 categories and 200 images in total.

2.2. PHOW

The Matlab's VLFeat library will be used for studying the problems proposed by Caltech101 and ImageNet. More specifically, the code phow_caltech101.m will be the base of this paper. The hyper-parameters that will be considered for the big databases are: train set size, test/val set size, number of words and spatial partitioning.

2.2.1 Understanding of PHOW

For a better understanding of PHOW, a list of little experiments will be performed. Those experiments will be performed in a tiny database derived from Caltech101. The goal of this process is to understand the effect of: the number of categories, the train and test sets size, the number of words, the C (SVM parameter) value, spatial partitioning, and others. This process will be the base of the predictions made in the rest of the paper.

2.3. Evaluating in Caltech101

The evaluation of Caltech101 will be performed taking into account a fixed number of images per category. This will be done in order to prevent misclassification due to probability (unbalanced classes). Each category will have 31 images (the minimum of images of a given category). The three most important hyper-parameters will be varied for obtaining an optimal performance. The values of the hyper-parameters will be found directly by testing in the test set.

2.4. Evaluating in ImageNet200

The evaluating process will be set in two parts. The first part will be the estimation of parameters and the second part will be the application of the model with the found parameters in the test set.

2.4.1 Estimation of parameters

The estimation of parameters will be done in the train set. This set will be split into a movable train and validation set. In this set the optimal values of the three most important hyper-parameters will be found.

2.4.2 Evaluation in test set

The optimal model found in the previous step will be applied to the test set. The ACA obtained will be reported with its corresponding time.

2.5. Comparison of results

The results obtained in Caltech101 and ImageNet will be compared. Patrons of misclassification will be identified and possible solutions to those problems will be proposed

3. Results

3.1. PHOW_Caltech101: Tiny data base

The default values of the function PHOW are: 5 categories, 15 images per category for train and the same number for set, 300 words per image, spatial X and Y equal to 2, size of 5 and step of 7, svm's *sdca* configuration and C equal to 10. Tables showing the effect of changes in these hyper-parameters is shown below. The middle row of the tables for each hyper-parameter corresponds to the default values. Variations in time between tables can be caused by other PC processes, however the ACA is the same.

Effect of changes in the number of categories

There are n-more images per each category added. The computational and time cost is higher. However, this is the obvious (and not the only) effect. Other effect is that as the number of categories increases, the possibility of misclassifying a given image also increases. This effect is seen in the ACA of table 1.

# of Categories	ACA	Time
2	96.67%	43 sec
5	92.00%	47 sec
10	83.33%	53 sec

Table 1. General effect of the number of categories in the ACA. (The rest of hyper-parameters are set to their default value).

To sum up, if the number of categories increases, the difficulty, the possibility of misclassification and the possibility of having a low ACA will also increase. Time of execution also increases as number of categories does the same (obvious). This factor is inherent to the problem (the number of categories of a problem is given, cannot be changed). In order to obtain acceptable ACAs in reasonable times the other hyper-parameters must be changed.

Effect of changes in the size of train set

If the size of train set augments (more images per category) it is expected that the results in test will be better. If the size of train set decreases the opposite result is expected. The previous assumption is based on that with a higher number of data in train the model will have higher variance and lower bias.

Train size	ACA	Time
5	82.67%	48 sec
15	92.00%	56 sec
30	93.30%	58 sec

Table 2. General effect of the number of images in the train set in the ACA. (There are 15 images in the test set and the rest of hyper-parameters are set to their default value).

The effect of changing the train size can be seen in table 2. The assumption of the relationship between number of images, bias and variance seems to be correct. However, it is important to note that the effect of the number of train images seems to be more significant when it is less than the train data, than when it is more than the train data. The time also tends to go higher as the number of train images increases (obvious). Theoretically, train size should be really significant in the time of computation, but this wasn't the case. One possibility is that other PC processes affected the time factor.

Effect of changes in the size of test set

The train set number of images' effect is expected to be similar in the train set. However, this is not the case as it is shown in Table 3. The number of test images seem to have less importance in the ACA obtained that the number of train images. This seems logical, as the model is built in the train set, not in the set test. However, it was expected that the ACA in 5 images would be higher than the ACA in 15 images, but it wasn't. This result is merely explained by luck (ACA will increase if the images that are being misclassified are taken out when reducing the test size).

Test size	ACA	Time
5	92.00%	62 sec
15	92.00%	65 sec
30	91.33%	72 sec

Table 3. General effect of the number of images in the test set in the ACA. (There are 15 images in the train set and the rest of hyper-parameters are set to their default value).

In Table 3 it was also observed that the ACA decreased when the test set was significantly bigger then the train set. However that decrease wasn't big enough to be significant. To sum up, changes in the test set have little effect on the ACA obtained. Additionally if the number of images in test increases, the time does the same (obviously).

Effect of changes in the number of dictionary words

It is expected that the ACA increases as the number of words increases. However, there is a risk of over-fitting that have to be taken into account. A decrease in the ACA and risk of under-fitting is expected when decreasing the number of dictionary words.

# of Words	ACA	Time
50	82.67%	33 sec
300	92%	59 sec
1000	97.33%	140 sec

Table 4. General effect of the number of dictionary words in the ACA. (The rest of hyper-parameters are set to their default value).

The previously mentioned expectations can be seen in Table 4. No over-fitting was observed, but the expected increase of the ACA was seen. It is important to say that this parameter seems to have the most significant importance in time. This parameter also seems to be the one that most significantly affects the ACA obtained.

Effect of changes in C (SVM parameter)

C is known as the coefficient of 'acceptance of misclassification'. Theoretically, if C increases the ACA should decrease. However, this effect couldn't be appreciated in Table 5. One possibility as that this parameter isn't of big importance in little databases. The time also doesn't seem to follow a pattern.

C	ACA	Time
1	92%	63 sec
10	92%	54 sec
50	92%	56 sec

Table 5. General effect of the value of C (SVM's parameter) in the ACA. (The rest of hyper-parameters are set to their default value).

The number of classes was increased to 20 for seeing if C had any significant effect in larger databases. The rest of the hyper-parameters were set to default.

C	ACA	Time
1	70%	86 sec
10	70.33%	93 sec
50	70%	89 sec

Table 6. General effect of the value of C (SVM's parameter) in the ACA in a database with more categories (20). (The rest of hyper-parameters are set to their default value).

In Table 6 similar results were obtained as in Table 5. The hyper-parameter C doesn't seem to have importance neither in ACA or in time. The optimal value of C seems to be determined empirically (with no clues), but the changes in the ACA are minimum.

Effect of changes in the spatial partitioning

This parameter was used for computing spatial histograms. These parameter defines the number of bins that will be used for describing the databases' images. This parameter can be a vector, but it has to be an ascending one. Maybe it is related with the partitioning of the image in [x,y] parts for its posterior histogram computation. If this is the case, this parameter is of huge importance for considering local spatial information. It's functioning is not clearly understood, but it's effect is shown in Table 7.

spatial X	spatial Y	ACA	Time
2	2	92%	51 sec
4	4	94.67%	50 sec
(2 4)	(2 4)	93.33%	57 sec
10	10	97.33%	68 sec
(2 10)	(2 10)	94.67%	71 sec
(4 10)	(4 10)	96%	72 sec
(2 4 10)	(2 4 10)	96%	69 sec

Table 7. General effect of the value of spatial partitioning X and Y in the ACA. (The rest of hyper-parameters are set to their default value).

As it can be seen in 7 the most significant changes were found between changes in 1D spatial partitions. The best method was found with the higher spatial value for X and Y (10). Anyway, the time consumption rose quickly as the values of 1D spatial partitions increased. 2D (two and three values) didn't showed improvements in performance than the values in 1D. This is one of the hyper-parameters that has a strong effect in the ACA obtained (not as strong as the number of words, but it is really significant).

Other parameters

There are other parameters like the type of solver of the svm ('sdca', 'sgd' and 'liblinear'). Each of the methods was proved with the default hyper-parameters and found the following results:

solver	ACA	Time
sdca	92%	55 sec
sgd	90.67%	57 sec
liblinear	NA%	NA

Table 8. General effect of the SVM solver method in the ACA. (The rest of hyper-parameters are set to their default value).

The method of liblinear showed an error. sgd performance was worse than sdca in both ACA and time.

Another parameter of interest was found in a possible syntax mistake in the Train Vocabulary part (Fig. 1). The default values for numTrain and numTest were 15 for both of them. The 30 might represent the total of images that will be taken into account, but that value was fixed to 30 instead of being expressed as the sum of numTrain and numTest.

```
% --------------------------------------------------------------------
%                                                     Train vocabulary
% --------------------------------------------------------------------

if ~exist(conf.vocabPath) || conf.clobber

  % Get some PHOW descriptors to train the dictionary
  selTrainFeats = vl_colsubset(selTrain, 30) ;      %Selects only 30 im
  descrs = {} ;
  %for ii = 1:length(selTrainFeats)
  parfor ii = 1:length(selTrainFeats)
    im = imread(fullfile(conf.calDir, images{selTrainFeats(ii)})) ;
    im = standarizeImage(im) ;      %Pasa a single, al parecer.
    [drop, descrs{ii}] = vl_phow(im, model.phowOpts{:}) ;       %descrs
  end
```

Figure 1. Possible syntax error in training the vocabulary.

To proof the mentioned hypothesis the change of syntax was made and the effect of changing the train set size was measured again (with default parameters). The results obtained were compared with those obtained in Table 2.

Train size	ACA	Time	Old ACA	Old Time
5	80%	46 sec	82.67%	48 sec
15	92%	54 sec	92%	56 sec
30	96%	68 sec	93%	58 sec

Table 9. General effect of the train set size in the ACA, after syntax correction in training the vocabulary. Old ACA and Time refers to the results obtained in 1. (There are 15 images in the test set and the rest of hyper-parameters are set to their default value).

As it can be seen in the Table 9, after correcting the syntax mistake the ACA was more sensible to changes in the size of train data. The good side of it is that these allows a better handling if this parameter for improving the ACA. The bad news is that the time complexity is also increased, so a balance between ACA and time must be performed.

There are other hyper-parameters inside the script as 'size' (corresponding to the scales at which the dense SIFT features are extracted) and 'step' (Step in pixels of the grid at which the dense SIFT features are extracted). For simplicity these two hyper-parameters will be set as default.

Finally, there are other hyper-parameters related to the creation of the SVM model like the maximum number of iterations in k-means which is set to 50 by default. This hyper-parameter won't be taken into account because of the high computational cost that it would take.

Optimization

For time saving, the resize process of the image was reduced to 100 rows instead of the default 480 rows. To test their

performance the ACA was calculated with different train size sets and the rest of parameters as default.

Train size	ACA	Time	Old ACA	Old Time
5	82.67%	11 sec	82.67%	48 sec
15	92%	16 sec	92%	56 sec
30	96%	21 sec	93%	58 sec

Table 10. General effect of images size in the ACA for different train sizes. Old Aca and Time refers to the results obtained in Table 1. (There are 15 images in the test set and the rest of hyper-parameters are set to their default value).

As it can be seen in Table 10, the time taken for evaluating the ACA was significantly reduced compared with the original resize value (480 rows). The ACA obtained was maintained and even improved for 30 images in the train set. To sum up, using reduced images can improve time performance without affecting the ACA obtained.

Summary

To sum up the most relevant hyper-parameters are the number of words, the spatial partitioning and the train set size. The train set size should be the first hyper-parameter to be fixed because of it's big relationship between ACA and time. The other two-hyper parameters should be optimized after an optimal train set sized has been identified. Image resize (to 100 rows) can be a good strategy for reducing computational and time cost.

3.2. Caltech 101 - Full Database

Consideration for testing

It is important to say that the categories in Caltech 101 are not balanced. To solve this problem and prevent classification by probability instead of feature descriptors a fixed number of images was taken into account. The minimum of images in a category is 31 images in the 'inline_skate', so this was the number of images taken into account per category. The number of images in the test set was defined as 31 - train size.

The train and test images per category were defined randomly to guarantee robustness of the model. This process was repeated only once for ensuring the reproducibility of the model.

Optimal train size

The train size hyper-parameter optimal value was firstly found by setting number of words to 100, spatial partitioning to 2 (same for X and Y) and resize of 480 and 100 rows. The results obtained can be found in Table 11 and 12.

Train size	ACA	Total Time	Time/image
5	43.14%	3min	7sec
7	45.92%	5min	12sec
10	50.00%	4min	12sec
12	51.86%	3min	10sec
15	54.47%	3min	14sec

Table 11. General effect of the train set size in the ACA in images resized to 480 rows. Number of words are equal to 100, spatial partitioning of X and Y is equal to 2. (The rest of hyper-parameters are set to their default value).

Train size	ACA	Total Time	Time/image
5	40.42%	42sec	1.6sec
7	42.52%	42sec	1.7sec
10	45.05%	58sec	2.8sec
12	46.39%	58sec	3.1sec
15	49.94%	24sec	1.5sec

Table 12. General effect of the train set size in the ACA in images resized to 100 rows. Number of words are equal to 100, spatial partitioning of X and Y is equal to 2. (The rest of hyper-parameters are set to their default value).

As it can be seen in Tables 11, the best result was obtained with 15 images per category in the train set (which means there are 16 images in the test set). This result corresponds to the expected one. It is logical that with more data in the train set, the model will have a higher variance and lower bias, thus it will produce better ACAs. The computational cost of having 480 rows (Table 11) is decent (approx. 11sec per image without significant variations with changes in the train size).

The performance of having images of 100 rows (Table 12) is lower than the obtained using 480 rows. This is logical, the bigger the data, the better the results (being below over-fitting). The computational cost associated with having images of 100 rows is extremely low (aprox. 2sec per image without significant variations between changes in the train size).

Because this database is quite small (after the balancing of classes consideration), the excess of time when using images with 480 rows is worth it. To sum up, the best train size for Caltech 101 is 15 images (each image with maximum 480 rows) per category.

Optimal number of words

In the *PHOW_Caltech101: Tiny data base* section it was demonstrated that the number of words had a big effect on the ACA obtained. However, the time was also strongly affected by this parameter. This hyper-parameter was changed in the best train size previously defined and the following results were obtained:

Train size	ACA	Total Time	Time/image
100	54.47%	3min	14sec
200	58.09%	2min	8sec
500	61.70%	4min	15sec
1000	64.34%	4min	16sec

Table 13. General effect of the number of words in the ACA in images resized to 480 rows and a train size of 15 images per category. Spatial partitioning of X and Y is equal to 2. (The rest of hyper-parameters are set to their default value).

Again, the expected result was obtained. As the number of words increases, the representation of the image will be richer (below over-fitting) and the ACA will be bigger. The time of execution didn't change significantly between changes in the number of words.

To sum up, the best number of words is obtained in 1000 words, in a train size of 15 images (each with 480 rows) per category. Higher number of words were not taken into account because it was considered that the risk of producing over-fitting was elevated.

Optimal spatial partitioning

The spatial partitioning had also a significant effect in the ACA in the previous section. Different to the number of words, this hyper-parameter wasn't very affected by time consumption. That is why this hyper-parameter's optimization was left at the end. Different values of the spatial partitioning were taken into account:

Train size	ACA	Total Time	Time/image
2	64.34%	3min	14sec
4	66.3%	3min	13sec
8	67.59%	5min	18sec
(2 4)	62.19%	3min	10sec
(2 8)	66.48%	6min	24sec
(4 8)	65.07%	6min	21sec
(2 4 8)	NA	NA	NA

Table 14. General effect of the spatial partitioning in the ACA in images resized to 480 rows, a train size of 15 images per category and 1000 words per image. Spatial partitioning of X and Y is equal to 2. (The rest of hyper-parameters are set to their default value).

The spatial partitioning of (2 4 8) ran out of memory. This suggests that this parameter's cost was higher than expected. Also it failed when the spatial partitioning was equal to 10. In general the time of computation rose as the spatial partitioning also increased. However the time per image was acceptable.

The optimal spatial partitioning was found at it equal to 8. The time difference between 4 and 8 was high, but it is considered that this excess of time is paid in an improvement in the ACA (+1.29%).

Best Method for Caltech 101

The best method for the database of Caltech101 was with 15 images per category in the train set, each image with a maximum of 480 rows, 1000 words per image, and spatial partitioning of 8. The performance descriptors can be seen in Table 15.

ACA	Total Time	Time/image
67.59%	5min	18sec

Table 15. Best method's performance in terms of ACA and time.

All of the hyper-parameters values were predicted in the previous section. The only surprise was found in the high computational cost found at high levels of spatial partitioning. The best method's confusion matrix can be seen in Fig. 2.

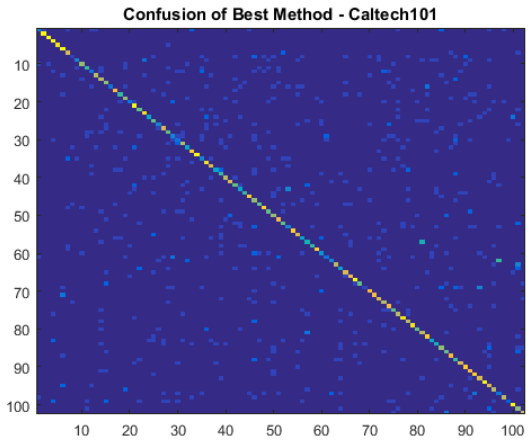

Figure 2. Confusion matrix of the best method for Caltech 101.

It can be seen that most of the data falls on the diagonal of the matrix (which means that the data is being correctly classified).

Common mistakes

One of the hardest categories to classify was the first one (BACKGROUND-Google). Personally I don't understand how this category was defined. It is hard for me as a human to understand what is the common pattern of this category. The huge intra-variety can be seen in Fig. 3.

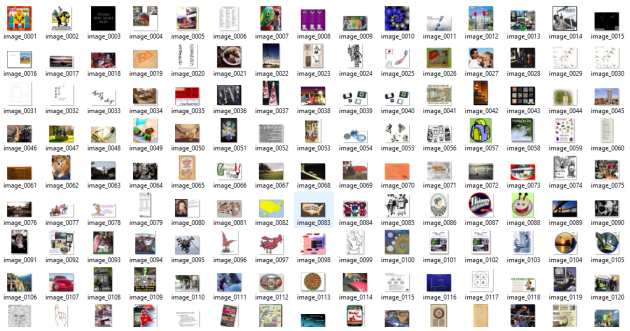

Figure 3. High variety of the BACKGROUND-Google category.

This category misclassification is somehow acceptable. For obtaining good results on this category a higher number of images in the train set can be considered. However, considering the huge variance of this category it would be hard to obtain good results.

Other two categories that were highly miss-classified were the little green dots out of the diagonal, corresponding to class number 57 and 62. These two classes correspond to 'ketch' and 'lotus'. The first category is composed by ships in the sea. The second one are lotus flowers in the lake. Both categories seem easy to classify (a white dot in a blue background and a pink point in a green background). However the difficulty of this classes is in the orientation of the images. Each image has a different orientation with different geometries in the background. That is a possible explanation of why does the method fails to classify these images.

Easy to classify classes

The easy to classify classes are those of low orientation variance. Examples of these can be seen in the categories of 'car_side' and 'Motorbikes'. Both categories always show a vehicle sideways. Considering that PHOW describes the form/geometry of images, these type of categories are really easy to classify.

Other factor that facilitates image classification is the constancy of the background (as in 'Faces_easy' and 'Leopards'). Harsh changes in the background can difficult the process of classification.

3.3. Image Net - Training

3.3.1 Considerations for training

For training the 'train' subset was used. This subset was divided in train and val for finding the optimal parameters. The hyper-parameters that will be taken into account are the same used in Caltech 101. To avoid redundancies, if the explanation of the results is the same as in Caltech 101 a 'same as in Caltech 101' will be shown. Additional explanations will only be given in cases in which the results obtained are not the expected (different from Caltech 101).

Train size will be changed again as the first hyper-parameter. Validation size will be 100 - train size. The other two hyper-parameters (number of words and spatial partitioning) will be changed later for obtaining an optimal ACA with an acceptable computational cost.

Optimal train size

As in Caltech101, the train size will be changed as well as the image dimensions (max rows equal to 100 or 256). It's result can be seen in Tables 16 and 17.

Train size	ACA	Total Time	Time/image
10	8.34%	18min	12.5sec
20	9.93%	18min	14.1sec
30	11.27%	18min	15.5sec
40	11.15%	18min	18.3sec
50	11.73%	12min	14.9sec

Table 16. General effect of the train set size in the ACA in images resized to 256 rows. Number of words are equal to 100, spatial partitioning of X and Y is equal to 2. (The rest of hyper-parameters are set to their default value).

Train size	ACA	Total Time	Time/image
10	5.71%	3min	2sec
20	6.67%	3min	2.5sec
30	7.24%	4min	3.3sec
40	7.34%	4min	3.9sec
50	7.14%	2min	2.6sec

Table 17. General effect of the train set size in the ACA in images resized to 100 rows. Number of words are equal to 100, spatial partitioning of X and Y is equal to 2. (The rest of hyper-parameters are set to their default value).

The best result was obtained in images of 256 and a train size of 50. However, some problems of over-fitting are seen in this database. When the train size is equal to 40, the model tends to over-fit. The same phenomena is observe in the images of 100 in the train size of 50. Again, the reduced method is way faster but produces significantly worse results (in terms of ACA).

Taking into account the probabilities of having over-fit, and seeing that there is no big difference between train size of 30-50 (in both Tables 16 and 17), a train size of 30 images will be defined as the optimal value. This optimal value (30 images per category in train set) will be evaluated initially in the 256 rows images, if problems of memory appear, images of 100 rows will be considered.

Optimal number of words

As in Caltech101, the next hyper-parameter to be defined is the number of words. The ranges used are the same ones used in Caltech101, and its results can be seen in Table 18

# of Words	ACA	Total Time	Time/image
100	11.27%	18min	12.5sec
200	13.5%	20min	16.9sec
500	16.84%	22min	18.4sec
1000	19.01%	13min	10.7sec

Table 18. General effect of the number of words in the ACA, in 30 images per category in the train size, with images resized to 256 rows. Spatial partitioning of X and Y is equal to 2. (The rest of hyper-parameters are set to their default value).

The maximum ACA was found at 1000 words per images (as in Caltech101). However, the most significant change in ACA was found between 200 and 500 words. The lesser increase from 500 to 1000 could be explained with over-fitting risk. The time behavior is not logical (more words should take more time). The excess of time can be explained because of multiple processes running at the same time. The high values of time present in 200 and 500 words can be caused by simultaneous run of these algorithms. The model for 100 and 1000 words were run by separated (only one run at a time), while the models for 200 and 500 words were run at the same time.

Taking into account that the most significant change was found between 200 and 500, the optimal number of words is equal to 500. This choice was also determined by the excess of time that having 1000 words would represent (and that the increase in the ACA is not worth it). 500 words seem to have a good balance between ACA and time performance.

Optimal spatial partitioning

As in Caltech101, the last hyper-parameter to be optimized is the spatial partitioning. The ranges used are the same ones used in Caltech101, and its results can be seen in Table 19

Spatial	ACA	Total Time	Time/image
2	16.84%	18min	12.5sec
4	18.01%	15min	9.1sec
(2 4)	19.65%	15min	9.1sec
8	NA%	NA	NA

Table 19. General effect of spatial partitioning in the ACA, in 30 images per category in the train size, images resized to 256 rows, and 500 words per image. (The rest of hyper-parameters are set to their default value).

The spatial partitioning equal to 8 showed the following error: 'Bad version or endian-key'. Apparently the error was present in the parfor of the SVM's training. This error is mainly produced because of memory overload when using parfors. This might make sense, since a higher number of spatial partitioning represents a higher number of partitions per image, which resolves in higher bins per histogram and higher computational cost.

Taking into account that the time between spatial partitions values was little, the best value for this hyper-parameter is (2 4). However, this hyper-parameter has a high risk of running out of memory for high values (4 partitions).

Best hyper-parameters for Image Net's training set

To sum up the previously mentioned optimizations, the best hyper-parameters considering ACA and time are 30 images per category, 500 words per image and a spatial partition of [2 4]. The ACA obtained for this parameters is of 19.65% and an estimated time of 9.1secs/image. However, a higher number of words (1000) can be considered for obtaining better results in terms of ACA.

3.4. Image Net - Test

Comparison between Image Net's train and test results

The SVM model was trained with the best hyper-parameters previously discussed in the train set and tested in Image Net's test subset. The method was both tested in 70 images per category that were never seen by the algorithm (as in the train set) and in 100 images per category. The method was tested using the previously defined model and train vocabulary. Each method computed spatial partitions for the whole subset. The results and the comparison between the train and test's subsets performances can be seen in the following table:

Subset	Imgs/category	ACA	Total Time	Time/image
Train	70	19.65%	11min	9.2sec
Test	70	19.40%	11min	9.5sec
Test	100	19.55%	11min	10.1sec

Table 20. Performance in train and test subset's with 70 or 100 images per category. Hyper-parameters: 30 images per category in the train size, images resized to 256 rows, 500 words per image and a spatial partition of (2 4). (The rest of hyper-parameters are set to their default value).

As it can be seen there are no significant changes of performance with variations in the test subset size. This suggests that the train model is significant enough to represent the whole database.

Best Method for ImageNet200

Summing up the previously discussed information, the best method for ImageNet200 is obtained with 30 images per category in the train size, images resized to 256 rows, 500 words per image and a spatial partition of (2 4). It's performance in the whole test set can be seen in Table 21.

# of Test Imgs	ACA	Total Time	Time/image
100	19.55%	11min	10.1sec

Table 21. Best method's ACA for test subset with 100 images per category. Hyper-parameters: 30 images per category in the train size, images resized to 256 rows, 500 words per image and a spatial partition of (2 4). (The rest of hyper-parameters are set to their default value).

The graphical representation of the best method's confusion matrix can be seen in Fig. 4.

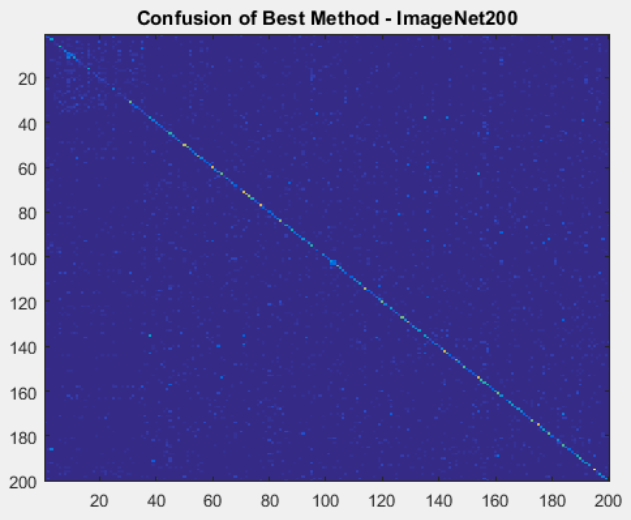

Figure 4. Confusion matrix of the best method for ImageNet200.

As it can be seen, the diagonal is less intense in ImageNet200, compared with the intensity obtained in Caltech101 (Fig. 2). This makes sense, as the ACA obtained in ImageNet200 is less than the one obtained in Caltech101.

Common mistakes and comparisons with Caltech 101

ImageNet200 has way more varied categories. In Caltech 101 most of the images were taken from catalogs, so most of the images were extremely easy to recognize. Most of them were taken from the same angle and with little variations in size. In ImageNet200 the images are taken from normal situations, not from catalogs. This causes that the intra-class variation is very high. The most common problems are related to occlusion (there is something in front of the object of interest), scale (different sizes for the same thing) and more than one object of interest per image (for example two weasels in the same image).

The best method proposed produced zero true positives in the category number 194, corresponding to the 'weasel' category. A small preview of this category can be seen in Fig. 5.

Figure 5. Weasel category of ImageNet200. Category that showed zero true positives.

As it can be seen this category has a high intra-class variety. It has weasels from all shapes, colors, orientations, backgrounds and sizes. It even has a raccoon in it. It is extremely difficult to find patterns in the images. It would be easy to think that a weasel should look like an elongated cylinder, but this isn't the case. Most of them are photos taken from the front of the weasel, which produces a variety of shapes between spheres and cylinders. However, it is surprising that this category showed 0 true positives. It is important to say that the classifications were spread trough different categories in a similar number (2-3). This suggests that the method didn't find any common pattern, and tried to classify by looking the category that looked more like it.

3.5. Further improvements

As it could be seen in some examples from Caltech-101, there are some categories that are easier to represent by its color information than by its shape information (as lotus flowers). It would be useful to run a sub-algorithm to see if it is easier to represent a category by color, by shape or by a combination of both of them.

Also, there might be a 'trash can' category (like the BACKGROUND_Google category). It would be useful to represent those images in a different dimension for preventing misclassification. Something like representing all of the previously mentioned points in a certain dimension, and the 'trash can' category in another one. This 'trash can' category can be thought as 'it doesn't look like anything in common' category.

For more complicated categories, as the ones from ImageNet200 it is harder to propose a solution. I would propose a Random Forest classification before doing the SVM's classification. Something to represent better the possible inter-class mistakes. For example, make a Random Forest for the orientation of the images. If the images are in a 'horizontal' orientation, draw them in a leaf and then run a SVM between all the horizontal images. This might solve the problem in which a certain image is classified because of its orientation but not its contents itself. The same could be applied for size and background information. However, it would be difficult to control that the Random forest actually sub-classifies by size, orientation and background information. A possibility can be obtaining texton's information for running Random Forests, and then run PHOW in SVM's.

The general idea of the Random Forest before SVM process is to sub-classify before classifying. First gather together things that look like them (ex. horizontal images) using textons and Random Forest, and then look for the differences between that using PHOW and SVM's.

An optimization process for considering if shape is important for representing a category, or if a color representation is better can be done. For example there are categories like 'white-wolf' in which a color representation should have more weight than the shape.

Another consideration is that PHOW takes by default the RGB2gray information. Maybe the Luminosity channel from Lab could give better results. Again, the color information should be taken into account.

4. Conclusions

- The results obtained in Caltech101 were better than the ones of ImageNet200. The main reason of this is that the categories were harder to classify, as the ImageNet200 images had way more variety than Caltech101's ones. Also, ImageNEt had more categories than Caltech101, which increases the possibilities of misclassification.
- PHOW is useful as a first approach for a classification problem. PHOW showed better results than luck for both Caltech101 (ACA of 67.59%) and ImageNet200 (ACA of 19.55%). This method showed a reasonable time cost per image for Caltech101 (18sec) and ImageNet200 (10.1sec). However, the memory needed for obtaining decent results with this method is high.
- PHOW is highly dependent of it's hyper-parameters. The most important hyper-parameters are train set size, number of words and spatial partition. Another important hyper-parameter for reducing the memory cost can be image downsampling. Finding the optimal hyper-parameters can be exhausting but the results obtained are decent.
- PHOW is not the best method for solving a classification problem. PHOW only takes into account shape information. It would be useful for it to take also into account color information for preventing misclassification.

References

[1] Khaligh-Razavi, S-M. What you need to know about the state-of-the-art computational models of object-vision: A tour through the models. Cambridge. UK.