Computer Vision 1: Final assigment

Jorn Ranzijn (11138610) Romeo Goosens (10424458)

March 2018

1 Part 1: Bag-of-Words based Image Classification

Using a bag-of-words approach for image classification, requires the following 5 steps.

1. Ferature Extraction and description.
2. Building a visual vocabulary
3. Quantize features using visual dictionary (encoding)
4. Representing images by frequencies of visual words
5. Classification

The following subsections will go into more detail about each step. Note that many of the above steps needs parameter tuning and design decisions, therefore the section 1.6 will explain these decision based on obtained results.

1.1 Feature extraction and description.

In this step features and the corresponding descriptors are collected from the training and test image set. The basic features are the SIFT features extracted from gray scale images but experiments have also been conducted with dense-SIFT and different color spaces. SIFT and dense-SIFT features were obtained from the following color spaces, the RGB color space, rgb color space, opponent color space and the $l_1l_2l_3$ color space. To obtain the descriptors in these color-spaces, an image is first converted to gray scale and features are extracted with either SIFT or dense-SIFT. This was done with the help of the *VL_FEAT* library, using the *vl_sift* and *vl_phow* functions, respectively. Next, the original image is transformed to the correct color space and for each channel SIFT or dense-SIFT is applied on the earlier determined frames. The obtained descriptors for every channel are then concatenated resulting in 384 dimensional vectors. The reason to use these specific color spaces is that they have varying invariances and usually an increase in the amount of invariances means a decrease in discriminative power for object classification. Therefore, it is useful to check if there exists a color space that strikes the right balance between these two forces. More specifically, it is useful to correctly classify images under varying image conditions and, in order do to so, invariant image features are needed. However, using a highly invariant color space might transform an image in such a way that too much discriminative power is lost which in turn decreases classification performance [2]. Table 1 shows the specific invariances that these color spaces have.

Table 1: Overview of the different feature extraction methods and their invariances

Feature Type	Light intensity change	Light intensity shift	Light intensity change and shift	Light color change	Light color change and shift
SIFT	+	+	+	+	+
RGB-SIFT	+	+	+	-	-
rgb-SIFT	+	+	+	+/-	+/-
Opponent-SIFT	+/-	+	+/-	+/-	+/-

Transformations to the specific color spaces are shown in equation 1, 2 and 3 for the rgb, opponent and $l_1l_2l_3$ color space respectively [4].

$$\begin{aligned} r &= \frac{R}{R+G+B} \\ g &= \frac{G}{R+G+B} \\ b &= \frac{B}{R+G+B} \end{aligned} \tag{1}$$

$$\begin{aligned} O_1 &= \frac{R-G}{\sqrt{2}} \\ O_2 &= \frac{R+G-2B}{\sqrt{6}} \\ O_3 &= \frac{R+G+B}{\sqrt{2}} \end{aligned} \tag{2}$$

$$\begin{aligned} l_1 &= \frac{(R-G)^2}{(R-G)^2+(R-B)^2+(G-B)^2} \\ l_2 &= \frac{(R-B)^2}{(R-G)^2+(R-B)^2+(G-B)^2} \\ l_3 &= \frac{(G-B)^2}{(R-G)^2+(R-B)^2+(G-B)^2} \end{aligned} \tag{3}$$

1.2 Building a visual vocabulary.

After feature extraction, a visual vocabulary was created using a subset of the training data. Only a subset was used as using the same data to train the classifiers leads to overfitting. Features were clustered using the K-means algorithm and the resulting clusters centers are referred to as visual words. These visual words help to describe images and allow us to classify images based on visual word distributions. Building the visual vocabulary requires tuning of two parameters. The first is the size of the subset of the training data and the second is the number of visual words/cluster centers. We think that increasing the amount of training data results in better visual features but this comes at a cost as there is less training data available to train the classifier. Finding a value optimal for the number of clusters is dependent on the number of features obtained after SIFT or dense-SIFT and the amount of training data used. It is desirable to have enough clusters so that the uniqueness of images can be expressed but it should not be too high as this would lead to sparse representations and overfitting.

1.3 Quantize features using Visual Vocabulary

After determining the visual words, we can assign the extracted features from the remaining training images and test images to these clusters. Extracted features are assigned to the visual word for which they have to lowest euclidean distance. Note that other distance metrics could be more applicable to determine visual words.

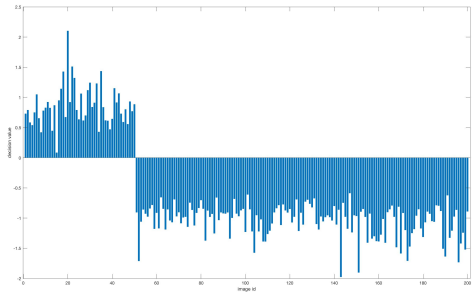
1.4 Representing images by frequencies of visual words

After obtaining the visual words for each image, the frequency of visual words for each image is determined and values are normalized. Normalization is needed due to the different amount of features for each image and therefore the number of visual words for each image varies.

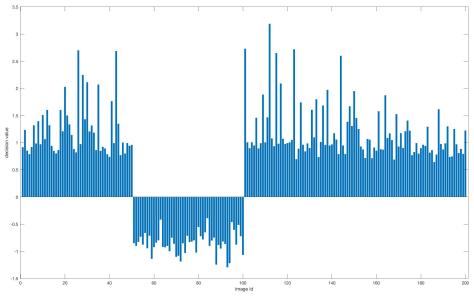
1.5 Classification

In the last step, a Support Vector Machine (SVM) is trained that takes the normalized visual word frequencies as an input and predicts the image class. Four binary classifiers are trained since there are four classes. A framework called *LIBLINEAR* is used to train the SVM [1]. After making predictions on the test data a HTML-file is generated for each image for every class and ordered by the largest distance to the decision boundary. Also, the corresponding average precisions for each class and mean average precision are determined.

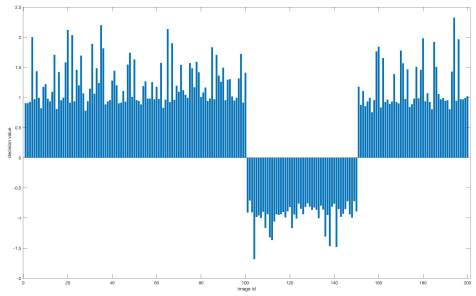
The LIBLINEAR library has the convention to assign the first datapoint to the positive side of the SVM decision boundary, meaning that the first data-point determines whether or not the predicted class gets a positive or negative value. When looking at figure 1a, 1b, 1c and 1d it is clear that only the first class (airplanes) get a positive value assignment and the other three classes gets a negative value, due to the fetched order of the data. To avoid this problem, predictions are made on the train data and it is checked if the mean value is positive or negative and decision values are changed sign if necessary.



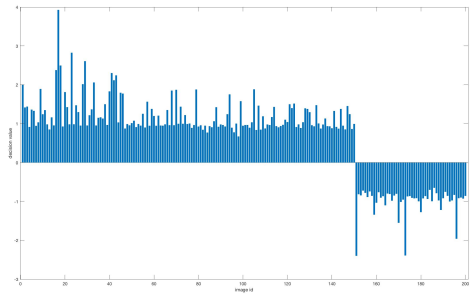
(a) Trainings data for class airplanes.



(b) Trainings data for class cars.



(c) Trainings data for class faces.



(d) Trainings data for class moterbikes.

1.6 Analysis and parameter Tuning

In this part, different parameter settings and methods for image classification using a visual-bag-of-words approach are tested. There are a number of parameters that require tuning in order to achieve the best performance. These are as follows: 1. The amount of training data used to train the SVM classifiers. 2. The amount of training data used to train the visual vocabulary. 3. The number of clusters/visual words in the visual vocabulary. 4. Feature extraction using SIFT or dense-SIFT. 5. The step size in dense-SIFT. 6. Transformation of the image to other color spaces, specifically, rgb, opponent and $l_1l_2l_3$.

To avoid an exhaustive grid search for the best parameter settings, an assumption will be made that all the parameters are independent of each other. This assumption, of course, is not true but we think that a reasonable approximation to the best solution can be found in this way and it reduces training time significantly.

New models are compared to a baseline in order to determine parameter value performance. The baseline was created with the following settings. 400 images from every class were used where half of the images are used to create the visual vocabulary and the other half is used for SVM training. Features were detected using regular SIFT and the number of clusters was set to 100. The training data for the faces class contains only 400 images, therefore, it was decided to take 400 images from every class as this would counter a class imbalance which could degrade performance.

1.6.1 Number of training examples

Table 2, shows the result when the number of training images are varied. Increasing the number of training example leads to a higher performance in general. Adding additional data that gives a better representation of the complete space of possible images should improve performance and thus this comes as expected. Interestingly, the best mean average precision is not obtained for the maximum amount of training data but for 200 training images per class. As there are only 200 images in total in the test set and the performance difference between 200 and 400 is fairly small, it was decided to continue with 400 images as the performance difference could very well be by chance. Cross-validation could provide more certainty.

Table 2: Number of training data images from every class

Number of training image for each class	mAP	accuracy
10	0,5777	0,4900
50	0,7642	0,73
100	0,8601	0,805
200	**0,898**	0,815
400	0,875	**0,815**

1.6.2 Fraction of training images for visual vocabulary creation

Another parameter is the fraction of training images that is used to create the visual vocabulary. Table 3 shows the result when this fraction is changed and the other parameters are kept the same as the baseline parameters. There does not seem to be a clear relationship between model performance and the number of images used to create the visual vocabulary. The hypothesis was that a trade-off is made between the quality of the visual words and the performance of the SVM classifier. However, the results do not really support this hypothesis as performance improves again when only $\frac{1}{12}$ of the data is used for visual vocabulary creation. Therefore, it was decided to continue with $\frac{1}{4}$ since this gave the best performance.

Table 3: Fraction of training data for k-means

Fraction	mAP	accuracy
1/2	0,875	0,815
1/4	**0,9162**	**0,84**
1/8	0,8703	0,795
1/12	0,8949	0,805

1.6.3 SIFT

Another version of SIFT is dense-SIFT, where at every k-steps a feature is determined of the image. This parameter is important since it significantly affects the accuracy, see table 4. It becomes clear that a higher density of features works better and that dense-SIFT outperforms regular SIFT. A downside of dense-SIFT is that the number of features from every images increases enormously, thereby increasing computation time. An almost 50 time increase in the number of features was observed when step size k is equal to 10.

Table 4: Dense SIFT

step size	mAP	accuracy
10	**0,9802**	0,945
20	0,9655	**0,95**
30	0,9712	0,915
40	0,9509	0,905

1.6.4 Number of visual words

Table 5 shows the performance when the number of visual words is varied. Increasing the number of clusters improves performance significantly indicating that a low number of clusters limits discriminating power. The advantage of a richer representation comes at the cost of extra computation time as K-means has to check how to assign features to higher amount of clusters. The best performance was obtained at 4000 clusters based on the mAP although the increase is low compared to result of 2000 clusters.

Table 5: Cluster size for k-means

Number of Clusters	mAP	accuracy
100	0,9393	0,865
400	0,9466	0,885
800	0,9400	0,88
2000	0,9702	**0,935**
4000	**0,9735**	0,92

1.6.5 Color spaces

Different color spaces were also tested. In turns out that the color spaces add additional discriminative power with the exception of the $l_1l_2l_3$ color space, see table 6. This color space is the most invariant color space tested and in turns out to be to strict as discriminative power is lost. It even performs worse than the baseline which applies SIFT on gray scale images. The best performing color space was the regular RGB. Here, we did not account for noise propagation in the unstable regions of the rgb color space as a low intensity is problematic. It might be that accounting for this improves performance.

Table 6: SIFT for different colorspaces

color space	mAP	accuracy
RGB	**0,9508**	0,875
rgb	0,9221	**0,88**
opponent	0,92556	0,865
l1l2l3	0,7123	0,715

1.6.6 Optimal model parameters

The optimal parameter settings and best design decision are shown in table 7.

Unfortunately, running this model turned out to be infeasible as the training time exceeded 23 hours and did not even finish at that time. It was decided to preemptively end the model. The increase in feature space by the RGB color space and increase in number of features by dense-SIFT and the amount of training data turned out to be too much. In order to reduce the training time, a new combination of features was chosen, see table 8. This reduced the training time to

Table 7: Optimal parameter settings

Parameter	Value
Training data	400 (per class)
Fraction for visual vocabulary	1/4
Dense-SIFT	step 10
Number of clusters	4000
Color space	RGB

approximately 90 minutes and resulted in a mAP of 0.9952 and accuracy of 0.9850. Reducing the number of visual words to 2000 seemed reasonable as performance difference between 4000 and 2000 was small. Also, the RGB color space increases the dimensionality of the features to 384 what greatly increased computation time for the K-means. Therefore, it was decided to use gray scale for SIFT as this results in 128 dimensional features and reduces training time. The number of features increases enormously with dense-SIFT but performance difference between a step size of 20 and 10 was fairly small which is why the step size was changed to 20.

Table 8: Optimal parameter settings

Parameter	Value
Training data	400 (per class)
Fraction for visual vocabulary	1/4
Dense-SIFT	step 20
Number of clusters	2000
Color space	Gray

1.7 Qualitative Analysis of best model

The results of the SVM classifier with the best parameters settings show a high mAP (0.99523). Looking at the individual classes, it is noticeable that the faces have an Average Precision of 1, meaning that all test images that had the label face, where classified correctly. Looking at the first 6 top images (figure 2) it is apparent that for the car class, almost identical images (number 1 and 2) are positioned directly under each other. Another important aspect is that the airplane image is for a human perspective not the best image of an airplane, we think this is due the multiple airplanes. From a human perspective the third, fifth and sixth image are more clear images off an airplane.

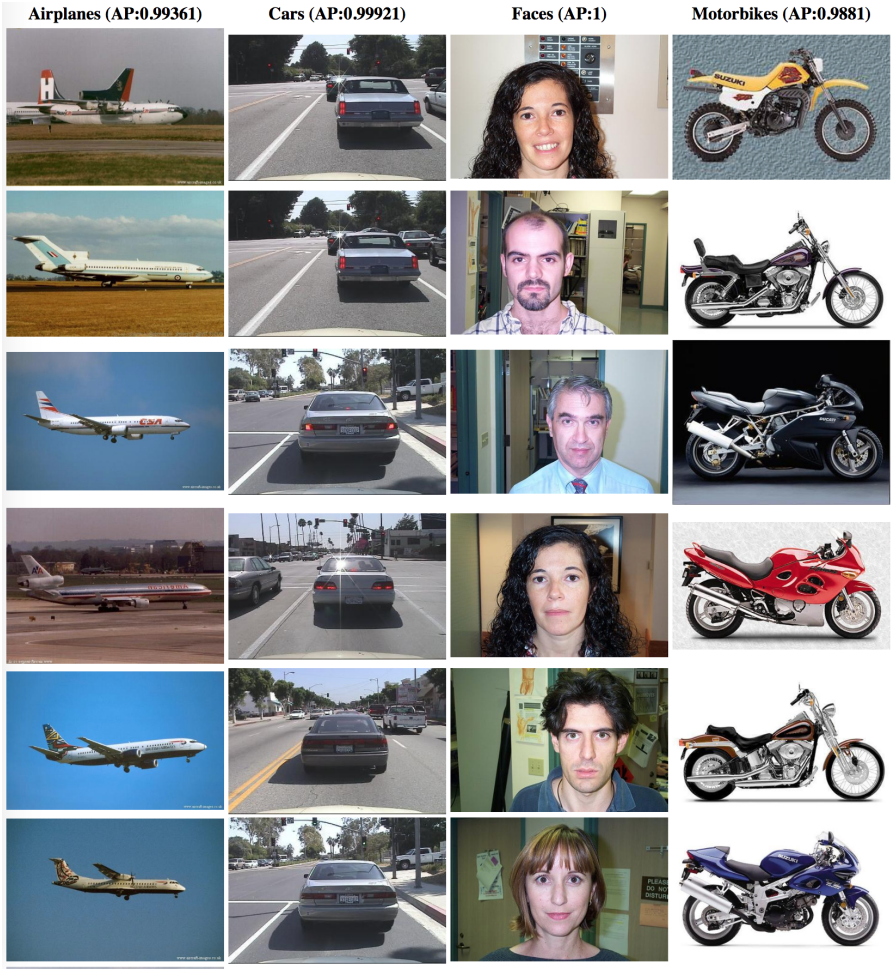


Figure 2: The first six ranks, from left to right the predicted classes are airplanes, cars, faces and motorbikes.

When look at rank 48 till 54 (figure 3), we see a few remarkable things. At rank 48 only airplanes has made one mistake so far (at rank 39) and motorbikes has made one mistake (also at rank 39). At rank 54 we see two fighter jets, these fighter jets were also wrongly classified in the motorbikes ranking. Due to the irregular shape, it is hard for classifiers to make the right prediction. Also, at rank 48 we see a motorbike and a parked car in the background, therefore, the classifier gives a high probability to this car. Looking at the motorbike images we see a very similar misclassification appear three times, namely the same black car . Although it is hard to tell where the classifier is "looking" at a reason for this classification could be that the *BMW* logo is trained from the motorbikes and therefore predict these cars around these ranks.

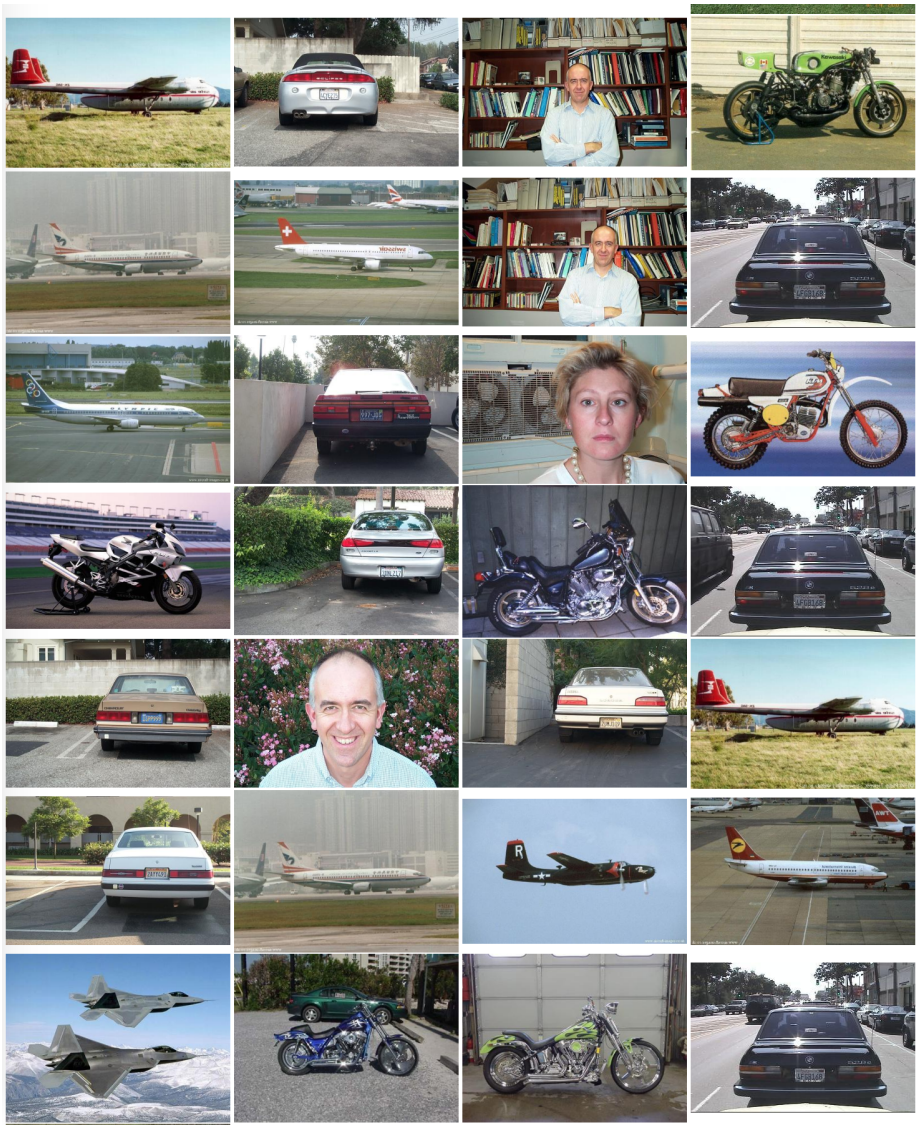


Figure 3: rank 48 till 54, from left to right the predicted classes are airplanes, cars, faces and motorbikes.

In the last quantitative analysis (figure 4), we looked at the last 20 rankings (ranking 180 to 200). A few remarkable things can be told about these final ranks, namely:

1. For Airplanes the number of faces and motorbikes is relatively high compared to the number of cars. Meaning that a car has a low prediction score compared to the faces and motorbikes.
2. For the cars we see many motorbikes, but for the motorbikes we do not see many cars. Also for the cars the number of faces is relatively low.
3. For the faces it is clear that we see many airplanes and a few motorbikes.
4. For the motorbikes we see many faces and airplanes but not cars.

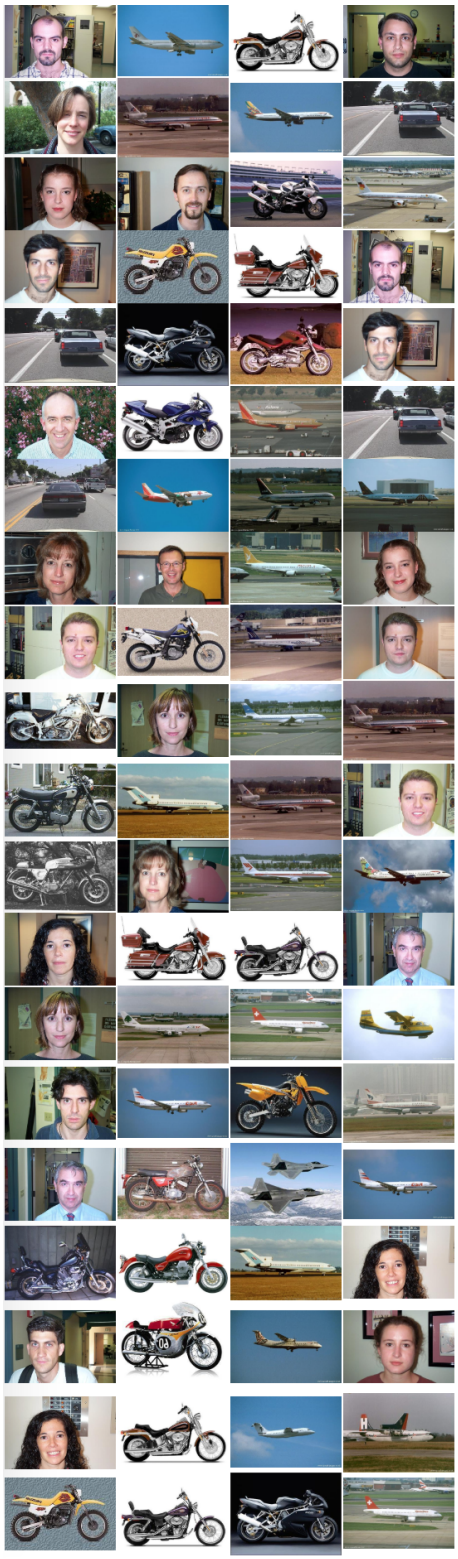


Figure 4: The last twenty ranks, from left to right the predicted classes are airplanes, cars, faces and motorbikes.

2 Part 2: Convolutional Neural Networks for Image Classification

2.1 Network Architecture

The pretrained network is a 13-layer neural network consisting of an input layer, 12 hidden layers and an output layer, see table 9 for the types of layers and corresponding dimensions.

Table 9: Architecture of the pretrained CNN.

Layer	Layer type	Dimensions (Width x Height x Depth)
1	Input	32x32x3
2	Convolution	32x32x32
3	Max pooling	16x16x32
4	Relu	16x16x32
5	Convolution	16x16x32
6	Relu	16x16x32
7	Average pooling	8x8x32
8	Convolution	8x8x64
9	Relu	8x8x64
10	Average pooling	4x4x64
11	Convolution	1x1x64
12	Relu	1x1x64
13	Convolution	1x1x10
14	Softmax	1x1x10

The first convolutional layer has the biggest size with $32 * 32 * 32 = 32768$ neurons and because of parameter sharing, the layer has only $32 * (5 * 5 * 3) + 32 = 2432$ weights. Layer 5, the second convolution layer, has $32 * (5 * 5 * 32) + 32 = 25632$ weights. Layer 8, the third convolution layer, has $64 * (5 * 5 * 32) + 64 = 51264$ weights. Layer 11, the fourth convolution layer has $64 * (4 * 4 * 64) + 64 = 65600$ weights. Layer 13, the last convolution layer has $10 * (1 * 1 * 64) + 10 = 640$ weights. Layer 11 has the highest number of weights.

2.2 Preparing the Input Data

The provided Convolutional Neural Network (CNN) requires a specific data structure as input. The data structure used for the CNN is called ***imdb*** consisting of an image component and a meta component. The image component consist of data (a 4D matrix of RGB-images), labels (the corresponding class of the image, in a range from [1,4]) and set (is the corresponding image a train (==1) or a test image (==2)). The meta component exists of all the side information about the classes and the splits. For the classes, list of strings are given where indices ([1,4]) correspond to the class names. The splits is also list of strings containing two labels: train and validation.

2.3 Updating the Network Architecture

Due to the relatively small amount of training data and long training times for CNN's, a pretrained CNN is used. This pretrained CNN is trained on the *CIFAR-10* dataset containing the labels: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The later layers are changed to four nodes in order to make the CNN suitable for classification of the four classes (airplane, car, face and moterbikes). Here, the last convolution layer (layer 13) and the softmax layer (layer 14) are changed to four output nodes where the weights of the entire CNN are trained gradually, meaning that the weights of the later layers are trained more heavily than the first layers which are "frozen".

Since the *CIFAR-10* and the *Caltech-4* have two of the same classes (airplane and car), we expect that the mAP for these same classes is higher compared to the other two classes (faces and motorbike), since the CNN is only partially trained for these last two classes.

2.4 Setting up the Hyperparameters

There are five hyperparameters to tune the CNN.

1. Learning rate previous layers. This is a parameter that determines how much we want to retrain the weights of the earlier layers in the network. Since we are using transfer-learning we are more interested in changing the new layers instead of the old ones.

2. Learning rate new layers. Parameter that determines how much we to retrain the new layers.

3. Weight decay is used to downscale weights. This is used to overcome high weights an it can be seen as a regularization parameter.

4. Batch size. The batch size is the number of images we can train the neural network in one forward/backward pass. Higher batchsize will result in a higher VRAM capacity, therefore this parameters is specific for the hardware used to train the CNN. All our experiments have been conducted using a *Geforce GTX 1070 8G* GPU to train the CNN.

5. Number of epochs. This parameters determines the number of forward and backwards passes of all the training data. Usually higher epochs results in a smaller training error, although this can result in overfitting.

2.5 Experiments

2.5.1 Feature Space Visualization

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for visualizing high-dimensional data. In the earlier described CNN model the last convolution layer is used as a feature representation for a SVM. This layer can be visualized by using the t-SNE technique. Figure 5 shows the corresponding t-SNE for the optimized parameter configuration (left) and the pretrained model (right). It is clearly visible that the optimized CNN has a very clear cluster/class separation compared to the pretrained model. Also, it is interesting to see that the cluster of faces (cluster 1) is more isolated from the other clusters, intuitively this makes sense since a airplane, car and motorbike have more similarities compared to a human face.

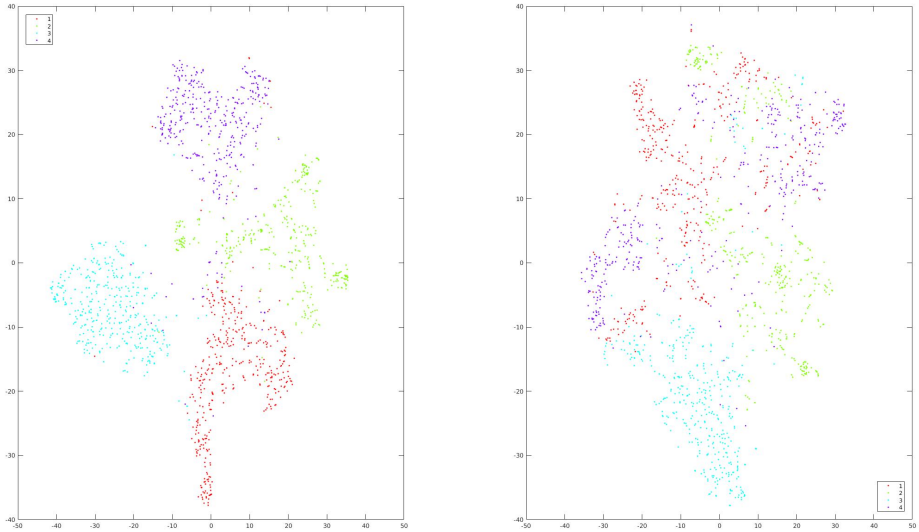


Figure 5: T-sne after 50 epochs for best model

Figure shows the t-SNE after only two epochs. Although the performance is relatively bad it is visible that there is "slightly" more cluster separation than the pretrained CNN.

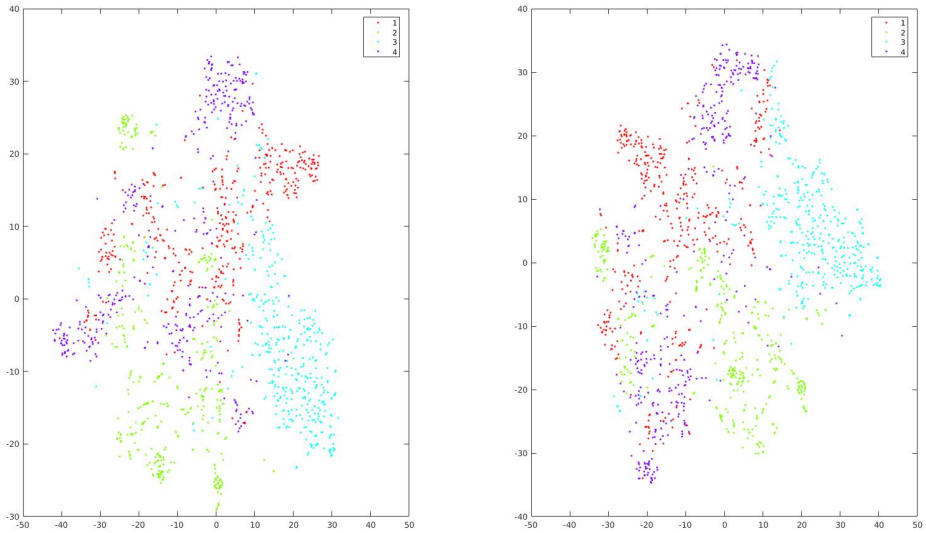


Figure 6: T-sne after only 2 epochs for the best model

2.5.2 Evaluating Accuracy

Experiments have been conducted to find the best set of hyperparameters. Different learning rates were tested for the old and new layers. It became clear that a better performance is obtained when only the new layers are retrained($lr\_new\_layer = 0.9$) and the previous layers are almost frozen

($lr\_prev\_layers = .002$). Changing the bias parameter does not have a big impact on the final performance of the CNN as well as the SVM.

To determine the batch size, multiple measurement have been conducted as show in table 10. It is shown that the best performance is obtained with a batch size of 400.

Table 10: Changing batch size

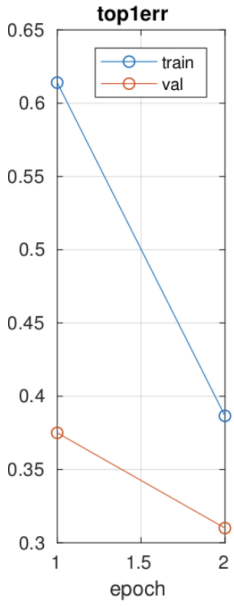
batch size	CNN fine tuned accuracy	SVM: pre trained accuracy	SVM fine tuned accuracy
50	0,92	0,9450	**0,97**
100	0,92	0,9450	0,9550
400	**0,93**	0,9450	**0,97**

The best epoch parameter is around 50 as shown in table 11

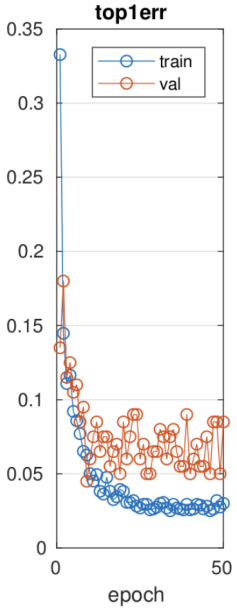
Table 11: Changing number of epochs

Epochs	CNN fine tuned accuracy	SVM: pre trained accuracy	SVM fine tuned accuracy
2	0,56	0,9450	0,9050
40	0,87	0,9450	0,9550
50	**0,93**	0,9450	**0,97**
80	0,78	0,9450	0,9600
120	0,89	0,9450	0,9550

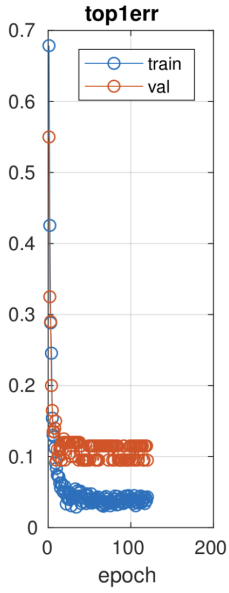
When looking at figure 7a, 7b and 7c the top 1 error is shown for multiple epoch steps. It is shown that when running too much epochs the model will overfit the data. This can been seen due decreasing error for the train data and the increasing error for the validation set (in Figure 7c.



(a) 2 epochs



(b) 50 epochs



(c) 120 epochs

Figure 7: Different top 1 errors.

3 Accuracy between CNN and BOW

The best Accuracy for the CNN is 97% whereas the best accuracy for the Bag-Of-Words is the $98,5\%$. Although the BOW is slightly better, both models where optimized with a limited amount of training time. Therefore, both models can be further optimized when more computation time is available.

Although both models have more or less the same accuracy, the CNN is more flexible when extending the data set or adding multiple classes. Also the CNN takes images as input whereas the BOW need sift features, this makes a significant difference in the computation time and scaling in the number of dimensions. The CNN is relatively faster to classify data which makes it more suitable for daily applications.

4 Discussion and conclusion

In this project, a visual bag-of-words in combination with a SVM and a CNN was used for image classification. For the bag-of-words approach, the assumption was made that every parameter could be optimized individually and then combined. Unfortunately, a model with the optimal parameter settings could not be obtained due to training time limitations. An approximation of these parameters was then used and achieved a near optimal accuracy and mAP. It seems likely that perfect scores can be achieved by either increasing the step size in dense sift or using a color space model such as rgb, RGB or the opponent color space. The CNN achieved similar scores as the visual bag-of-words approach. A pretrained network was used where weights in the latter layers were retrained more heavily compared to the earlier layers. The learning rates for these layers were obtained using a trial and error approach and can be fine-tuned even more to obtain a better performance. The main advantages of the CNN over the visual bag-of-words approach is that training time is shorter when transfer learning is used. Also, images features do not have to be extracted which saves a lot of hyperparameter tuning. We think that one advantage of the visual bag-of-words is that variable image sizes can be included more easily as histogram normalization makes it invariant to this problem. For CNN's, rescaling is needed and details might be lost. Further optimisations could be made to the CNN by training with augmented data or to use a boosting method such as drop-out in order to reduce overfitting [3].

References

[1] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. Liblinear: A library for large linear classification. *J. Mach. Learn. Res. 9* (June 2008), 1871–1874.

[2] Gevers, T., Gijsenij, A., van de Weijer, J., and Geusebroek, J.-M. *Color in Computer Vision: Fundamentals and Applications*, 1st ed. Wiley Publishing, 2012.

[3] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[4] van de Sande, K., Gevers, T., and Snoek, C. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 9 (Sept 2010), 1582–1596.