Computer Vision 1 - Final Project Part 1
Bag-of-Words based Image Classification

Wolf, Florian Biertimpel, David Lindt, Alexandra Fijen, Lucas

October 5, 2019

Introduction

In the first part of our final project we explore the task of image classification using a Bag-of-Words (BoW) model. For this purpose, we implement a BoW model and examine its classification performance qualitatively and quantitatively for various settings. These settings include two different approaches to sampling interest points from images (keypoint-based and grid-based) and three different sizes of the visual vocabulary used for transforming images into a histogram presentation. We further explore how well our system performs dependent on the color space in which its input images are represented.

1 Bag-of-Words based Image Classification

The Bag-of-Words (BoW) approach to image classification is based on representing an image as an unsorted collection (i.e. *bag*) of the features it contains. For example, an image of a flying plane would most likely contain a number of edges and corners of a grey-ish or white-ish object on blue background. Such characteristic features can be extracted from an image and used to build a more information-dense representation for it, the BoW representation. This representation can then be used to train classifier models, which in our specific case are binary Support Vector Machines (SVMs) [1].

As training data for our models, we use images from the STL-10 data set [2]. This data set contains images of ten different object classes. However, we only make use of the images of five classes, namely *airplanes*, *birds*, *cars*, *horses* and *ships*. All images are of size $96 \times 96 \times 3$. We use 2500 images for training and 4000 images for testing with images of all five classes appearing in equal parts.

We train our model with images represented in three different color spaces, namely *gray*, *RGB* and *opponent*. The image color space can be considered a hyper-parameter of our model. Figure **??** shows an example image from the dataset represented in the different color spaces. In figure 1b the image is converted to the gray scale color space, where it is represented by a single layer in which a low pixel value is dark, and a high value is white. Figure 1c shows the image represented in RGB color space: Three channels are of which one shows the amount of red, one the amount of green and one the amount of blue in the image. Finally, 1d shows the image in the opponent color space. Within this color space each channel represents the difference between two colors: In the first channel a high value represents a red color and a low value a green color, in the second channel a high value decodes yellow and a low value blue and in the third channel a high value means bright, and a low value a dark part of the image.

(a) Original (b) Grayscale (c) RGB (d) Opponent

Figure 1: Example image in different color spaces.

1.1 Feature Extraction and Description

For extracting representative features from images, we make use of SIFT descriptors []. These descriptors characterize an image by representing gradients contained in the neighborhoods of certain pixels. In order to decide for which pixels of an image the descriptors are calculated, we consider two different approaches. The *key point based* SIFT approach calculates the descriptors at points of interest within an image, such as edges, corners or blobs. In this way, we consider only (supposedly) characteristic points of an image. However, there is also no guarantee that we will not miss out on any important areas of an image. In contrast, the *dense* SIFT approach calculates the descriptors for a grid of pixels within an image. The advantage of this is that we are guaranteed not to omit any potentially characteristic areas of an image. However, we also have significantly more descriptors per image, which increases the computational effort. For the implementation of both approaches we make use of the functions `vl_sift` and `vl_dsift` provided by the MatLab library `VLFeat` [3]. Note that we perform dense SIFT with the `VLFeat` recommended default parameters of 8 for step size between evaluated pixels and 3 for the size of the considered spatial bin.

Additionally to examining the BoW model performance for different ways of extracting descriptors from an image, we want to investigate the impact of our data being represented in different color spaces. For this purpose, we train our model on images represented in gray scale, RGB and opponent colors. Since the `vl_sift` and `vl_dsift` operate on gray scale images, we use a workaround in order to profit from using colored input images: We apply the feature extraction methods on each color channel separately and then concatenate the found descriptors of all three channels to one vector. Note that we wish to describe the areas around the same points in different channels. For `vl_dsift`, the channel descriptors consider the same pixels per definition. However, the key point based `vl_sift` may find different key points for different channels. Therefore, we first transform our three channel image to gray scale, apply `vl_sift` on this gray scale version to find the overall key points, calculate the descriptors at these key points for all three channels, and finally append the descriptors of all channels into one. For `vl_dsift`, we are also performing a smoothing on the image previous to the feature extraction, as suggested in the tutorial provided by `VLFeat`.

1.2 Building Visual Vocabulary

Of our total 2500 training images we use a third (= 830 images) to build up our visual vocabulary. This vocabulary can be regarded as the collection of the most common descriptors among a set of images. In order to define this collection, we first extract all feature descriptors from our subset of training images. Then, we perform the clustering algorithm *kmeans* on all extracted descriptors. For running *kmeans* we need to set the number of clusters that we want to divide the retrieved descriptors into. This number is considered a hyper-parameter of our model, we test it with the values 400, 1000 and 4000. Running *kmeans* on our descriptors gives us our target number of cluster means. Each of these cluster means is considered a 'word' of our visual vocabulary.

1.3 Encoding Features Using Visual Vocabulary

With our visual vocabulary defined, we can now translate an image into a *histogram representation*. This representation is a vector of the same size as the number of descriptors in our visual vocabulary (i.e. 400, 1000 or 4000). For a given image, we calculate all SIFT descriptors using either the dense or the key point based approach. Then, we compare every descriptor we found to the descriptors in our visual vocabulary in order to find the most similar one. Once found, we count the the most similar descriptor up by adding 1 to the respective position within our histogram vector. Finally, we have to normalize the histogram vector, because we might find different amounts of descriptors for different images in case that we use key point based SIFT descriptors.

1.4 Representing images by frequencies of visual words

Before we start training SVM classifiers with the remaining training images, we first translate all images into their more compact histogram representations with the method described above. The histograms contain the information given by an image, but in a more *dense* form (i.e. as a vector of size 400/1000/4000 instead of an image of size $96 \times 96 \times 3 = 27648$) and are therefore easier for an SVM classifier to distinguish.

1.5 Classification

Since we use a third of our training images to build up our visual vocabulary, we have two thirds (=1670 images) left to train our SVM classifiers. In total we train five binary SVM classifiers, one for each image class present in our training data. Due to limits in available computational power, we decide to only use half of the images (= 835 images) for the training of each classifier. These 835 images are randomly sampled from the original 1670 images for every classifier individually. Note that we take the same amount of images per image class, which results in each binary classifier getting 167 positive and $4 \times 167 = 668$ negative examples. We label the positive data points 1 and the negative data points 0 respectively.

After preparing the input data, we can finally train the classifier models. For this purpose, we use the `MatLab` implementation of SVM, namely the function `fitcsvm`. We configure the classifiers to have a Radial Basis Function (RBF) as kernel function. Since the hyperplane formed by an SVM can be negatively influenced by the scale of the input features [1], we standardize our training data (mean 0 and variance 1) before training. As already mentioned above, the training data for each SVM is very unbalanced: We have four times as many negative examples as positive ones. To prevent our SVMs from just classifying every image as negative, we have to scale our kernel (i.e. adjust the σ of our kernel RBF) [1]. For this purpose, we make use of the `KernelScale` property of the `fitcsvm` function.

1.6 Evaluation

After training our five binary SVM classifiers, we apply each of them to our 4000 test images. For each image and classifier, we obtain the class that the classifier assigns to the image (i.e. 0 or 1) as well as the corresponding *positive class score*, which represents *how positive* the classifier estimates the image to be. This score is positive for images that have been classified as positive (i.e. as 1) and negative for the ones classified as negative (i.e. as 0). The higher the *positive class score* of an image assigned by a classifier is, the more the classifier considers the image as positive example and the lower the score is the more it is considered a negative one.

To examine the performance of our BOW based image classification model, we perform a qualitative and quantitative evaluation. For the qualitative evaluation, we look for each binary classifier at the *positive class scores* it outputs for our test images. Then we rank these scores and look at the images with the five highest and the images with the five lowest scores. These images are the ones that the respective classifier estimates to be the most positive/negative examples in our training data, i.e. the most/least representative images for the class that the respective classifier considers positive. To exemplify, if we do this using the classifier for the class *airplanes*, we will obtain the 5 images that our classifier considers the most *airplane-like* and the 5 that it considers the most *airplane-unlike*. For the quantitative evaluation, we calculate the average precision for each of the classifiers. To further make the performance of the BOW based image classification comparable to the performance of the CNN implemented in the second part of this project, we calculate the system's mean accuracy.

2 Results

In order to examine the performance of our BoW based image classification model for different hyperparameters, we train the overall model (consisting of five binary SVM classifiers) for a total of 18 settings: One for each possible combination of image color spaces, SIFT types and vocabulary size. Following, we calculate for every model the average precisions (AP) per class, the mean average precision (MAP) and the mean accuracy of all five SVM classifiers. We further look for every model for each of the SVM classifiers at the images with the highest and lowest *positive class score* to see which images they consider to be the most and least representative for the respective class.

2.1 Quantitative Results

Table 1 displays the calculated APs for the ranked results (from most certain to least certain given a class) and MAPs for the BoW model trained with different hyper-parameter settings. The highest scores per column are highlighted in yellow. Table 2 shows the accuracies achieved for the different hyper parameter settings. It is clear that accuracy-wise the classifiers overall performed a lot closer to each other. The overall method with the highest accuracy and MAP was the one that represents images in opponent color space, extracts feature with the dense SIFT approach and uses a visual vocabulary size of 400.

2.1.1 SIFT types

When comparing the performances of the models that use regular SIFT for feature extraction to the ones that use dense SIFT, it is clear that the ladder perform better in MAP and accuracy. The explanation for this is probably that using dense SIFT leads more features to be extracted from an image and therefore to more representative histogram representations of the images.

Colorscale	SIFT Type	Vocabulary Size	MAP Overall	AP Planes	AP Birds	AP Cars	AP Horses	AP Boats
Gray	**Regular**	**400**	0.482	0.529	0.334	0.513	0.499	0.537
		1000	0.460	0.491	0.327	0.508	0.474	0.502
		4000	0.358	0.414	0.260	0.442	0.255	0.423
	Dense	**400**	0.749	0.686	0.767	0.739	0.834	0.717
		1000	0.749	0.711	0.753	0.748	0.819	0.712
		4000	0.734	0.715	0.722	0.747	0.787	0.697
RGB	**Regular**	**400**	0.499	0.520	0.345	0.559	0.550	0.521
		1000	0.466	0.518	0.335	0.485	0519	0.483
		4000	0.425	0.452	0.295	0.493	0.426	0.459
	Dense	**400**	0.766	0.704	0.780	0.763	0.856	0.729
		1000	0.775	0.720	0.774	0.781	0.831	0.770
		4000	0.723	0.695	0.610	0.767	0.823	0.718
Opponent	**Regular**	**400**	0.527	0.562	0.377	0.573	0.580	0.541
		1000	0.501	0.534	0.385	0.526	0.563	0.500
		4000	0.457	0.499	0.269	0.537	0.528	0.452
	Dense	**400**	0.782	0.718	0.773	0.830	0.827	0.760
		1000	0.777	0.705	0.767	0.815	0.831	0.768
		4000	0.727	0.641	0.700	0.802	0.759	0.734

Table 1: Average Precisions per class and Mean Average Precisions of models with different hyper parameter settings.

Colorscale	SIFT Type	Vocabulary Size	Classifier Accuracy
Gray	**Regular**	**400**	0.808
		1000	0.802
		4000	0.8
	Dense	**400**	0.869
		1000	0.858
		4000	0.813
RGB	**Regular**	**400**	0.810
		1000	0.803
		4000	0.8
	Dense	**400**	0.874
		1000	0.865
		4000	0.813
Opponent	**Regular**	**400**	0.816
		1000	0.805
		4000	0.8
	Dense	**400**	0.881
		1000	0.8732
		4000	0.82655

Table 2: Accuracies of models with different hyper parameter settings.

2.1.2 Color Spaces

We see that the lowest scores are achieved using the gray color space, followed by the RGB space. Using the opponent color space overall leads to the highest MAP. However, when we look at the models' AP scores for single classes we see that images of some classes were slightly better classified by models that used the RGB color space. It does intuitively make sense that the gray scale performs worse than the spaces that consider colors, as one could imagine that in some cases color can be a useful method for detecting certain objects. As examples we can think of a blue sky for planes, green grass for horses or brightly colored cars.

Interestingly the models using the opponent space performs better in detecting cars, while all the other classes had their highest AP with the model using the RGB color space and dense SIFT feature extraction. As the opponent space is better at making a clear distinction in values between green to red, and yellow to blue, possibly this difference in performance is caused by some brightly colored cars that could in opponent space be better distinguished from the mostly blue airplane and ship images as well as from the mostly green/brown-ish horse and bird images.

2.1.3 Vocabulary size

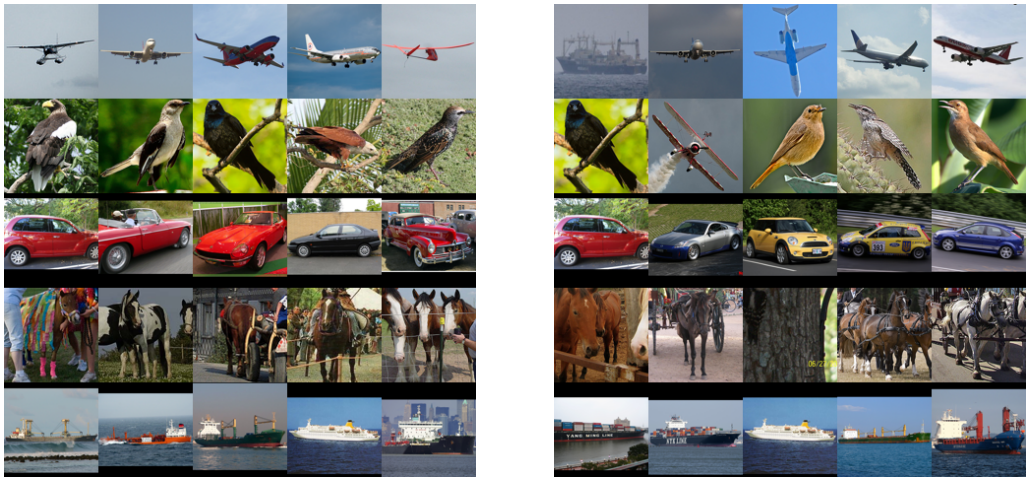
Another interesting point was that for the models with regular SIFT feature extraction the smallest vocabulary size (= the smallest amount of clusters in the k-means clustering) performed best. At the same time, for each SIFT and color space combination the models with a vocabulary size of 4000 performed worst. A possible explanation for this could be that using 4000 cluster centers creates very small clusters that are not a general enough, i.e. do not represent the most common extracted features. Small clusters could also result in very sparse histograms, which are more difficult to classify with a SVM. For the models with dense SIFT feature extraction, there was no big difference in performance between using a vocabulary size of 400 and 1000. This does fit into the picture: Since dense SIFT extracts more features than key point based SIFT we can use more cluster centers and still generalize enough.

2.2 Qualitative Results

In addition to our quantitative evaluation, we also illustrate for every model for each of the SVM classifiers the images with the highest and lowest *positive class score*. These are the images that the models' classifiers consider to be the most and least representative for the respective class. For each model, we shows these images in two figures: Each row in an figure represents a class. From top to bottom the rows the classes are: *planes, birds, cars, horses* and *boats*. One figure shows the top five images (i.e. the images with the highest positive class score) and the bottom five images (i.e. the images with the lowest positive class score). We include only some graphics in the results part, however, all of them can be found in the appendix.

Figure 2a shows the result for the model using opponent color space, a dense SIFT feature extraction and a vocabulary size of 1000. This model is one of the overall top scoring ones. We can see that the top five pictures for each class consist of only true positives. It is also remarkable that there are several pictures with very similarly strong colors, especially for the birds and the cars. As the color space was opponent, we think that the reason for this is that the channels in this color space primarily distinguish colors. We see that the model is especially confident on classifying red cars as cars and birds with green background as birds.

In figure 2b both the ship and the bird classes show a false positive. For the bird class we see that a diagonal plane is labeled as a bird. We saw this happen in several other models as well. In multiple images, birds as well as planes have a strong diagonal component. In this case, the model picks yellow birds with a strong diagonal component. The mistaken airplane has such a diagonal component



(a) (b)

Figure 2: Top results opponent dense SIFT: left vocabulary size 1000, right 4000

and additionally fits color-wise. We observe this mistake also the other way around in several cases. Especially bird images with a blue background color are frequently mistaken for airplanes.

In figure 2b, a ship is confidently labeled as a plane. For both the ships and the planes we often observe that there is some strong horizontal component with a gray/blue-ish background. Therefore, we can easily imagine why a classifier would confuse them. Additionally, we frequently see that the ship images that contain some strong whites combined with blobs of strong colors. In contrast, the image of the ship that is falsely labeled as a plane contains neither white or strong color blobs but is instead of an overall gray-ish color.

From the table 1 we could see that cars and birds have a higher average precision than the other three image categories. This could make sense as these two categories don't seem to have as many shared components with images of other classes and appear rather unique in most of the cases.

As expected and evident in the appendix, the images with the lowest positive class scores seem randomly selected among the image classes. The kind of images that are seen very much differ between the models. In several, planes or ships are overly represented.

Conclusion

In this part of our final project, we explored image classification with BoW models. We analyzed the influence of representing our train and test images in different color spaces, namely gray scale, RGB and opponent color space. We found that using of RGB and opponent color spaces outperformed using the gray color space. Further, we saw that BoW models trained on opponent space images have a tendency to consider images with strong colors as more representative for single classes , whereas the RGB based models seemed to be less selective on colors. We further experimented with two different approaches to feature extraction: dense SIFT and key point based SIFT. Overall we found that dense SIFT extraction strongly outperformed the regular SIFT extraction. Finally, we tried different size for our visual vocabulary. For the regular SIFT types a size of 400 lead to the best performance, whereas for the dense SIFT types 400 and 1000 worked just as well. We further found that a vocabulary size of 4000 is too big and leads to a decrease in performance. Our best overall approach was an opponent color space based model using dense SIFT extraction and a vocabulary size of 400. We scored an accuracy of 0.881 and a mean average precision of 0.782 with this particular model. Finally, we found that among our five image classes, cars and horses were much easier to classify than planes, birds and

boats.

References

[1] Asa Ben-Hur and Jason Weston. A user's guide to support vector machines. In *Data mining techniques for the life sciences*, pages 223–239. Springer, 2010.

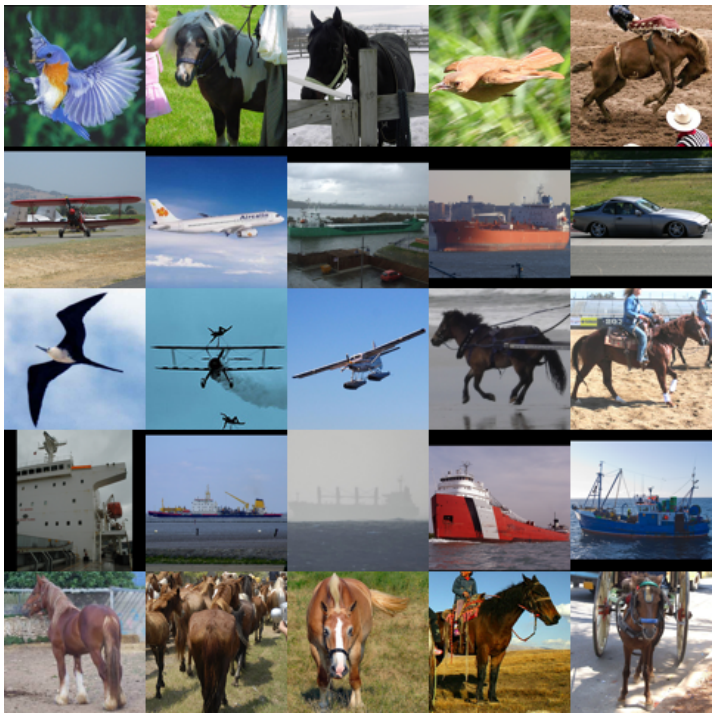
[2] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

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

Appendix

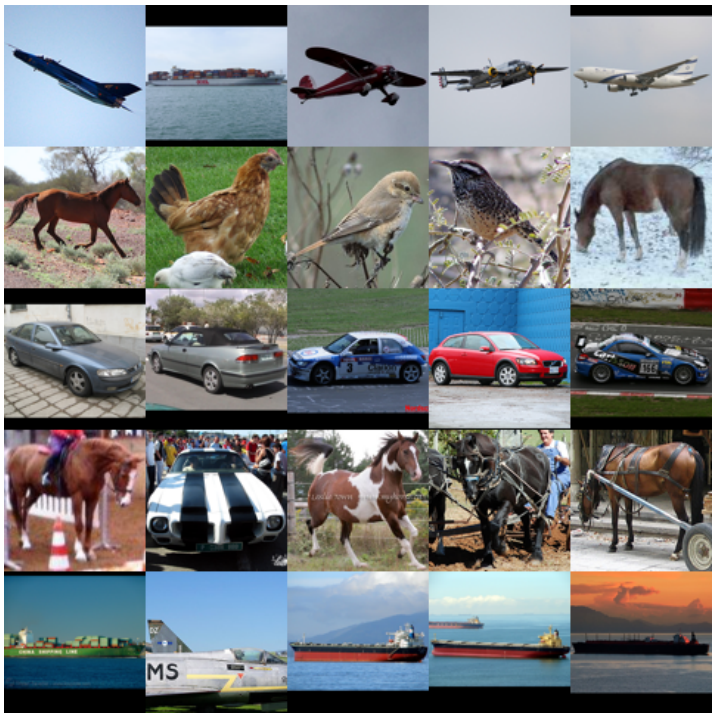
Figure 3: Grayscale regular SIFT with 400 vocabulary size

(a) Top 5 most certain

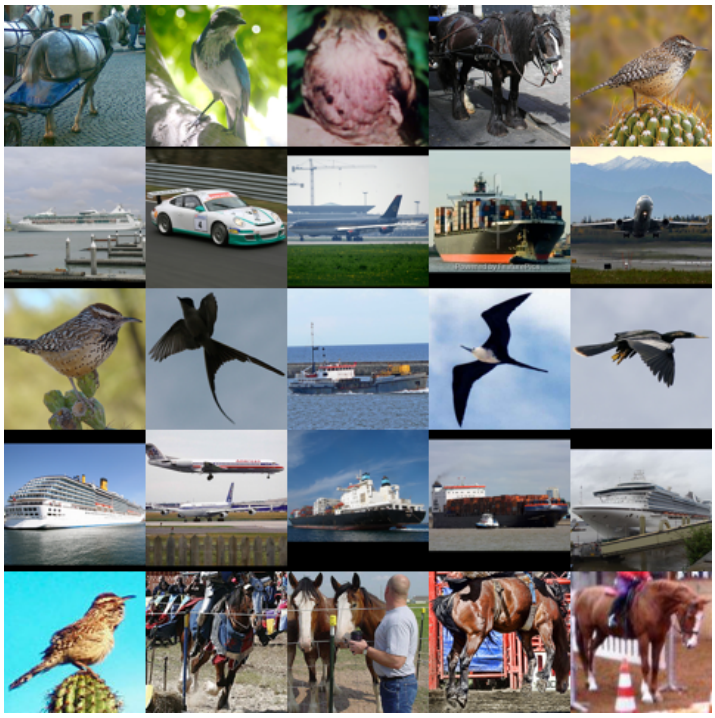


(b) Top 5 least certain

Figure 4: Grayscale regular SIFT with 1000 vocabulary size

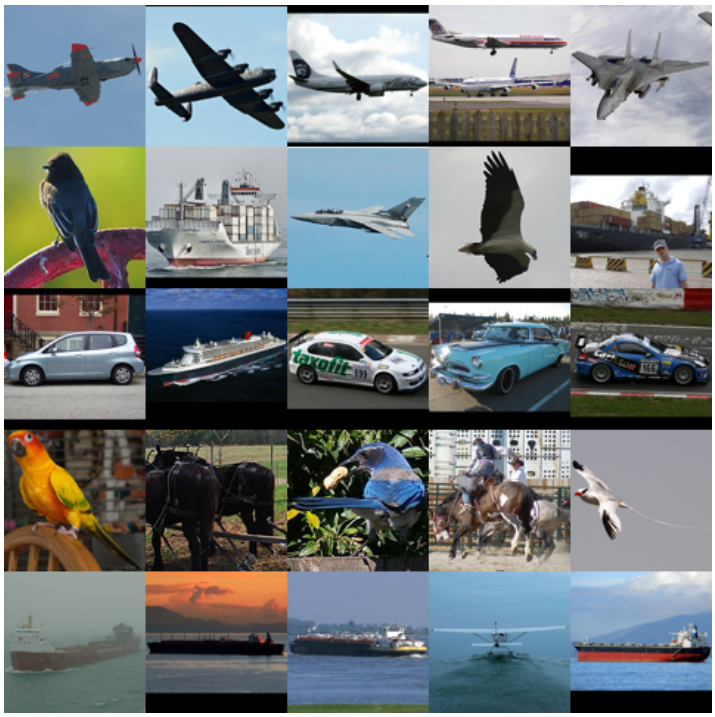


(a) Top 5 most certain

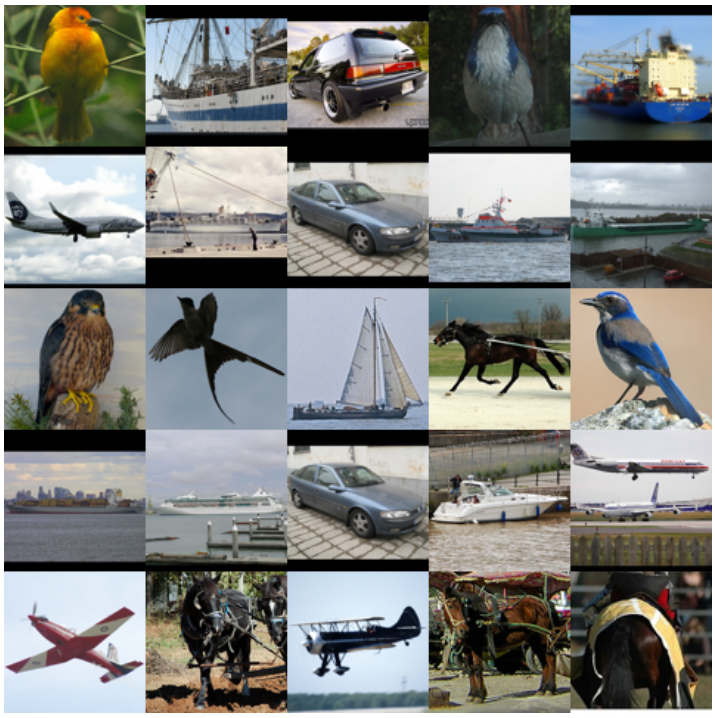


(b) Top 5 least certain

Figure 5: Grayscale regular SIFT with 4000 vocabulary size

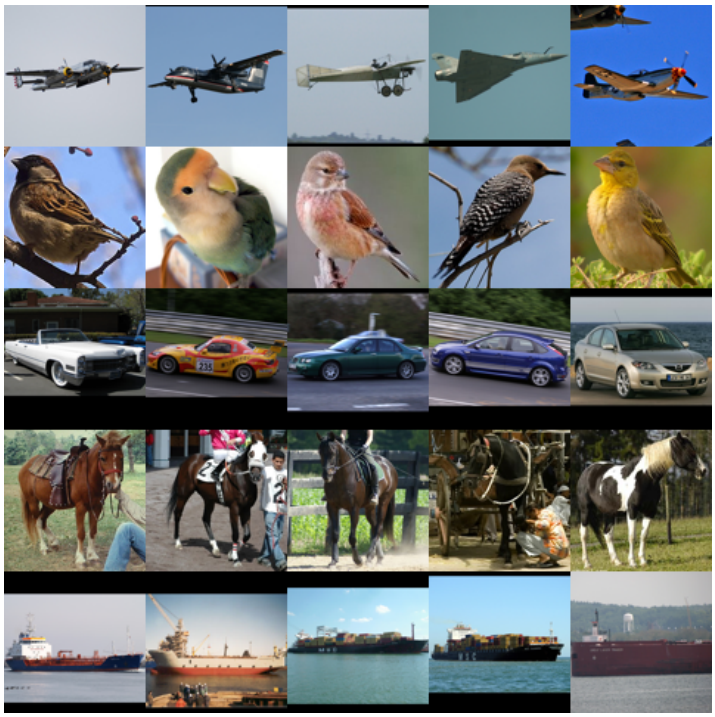


(a) Top 5 most certain

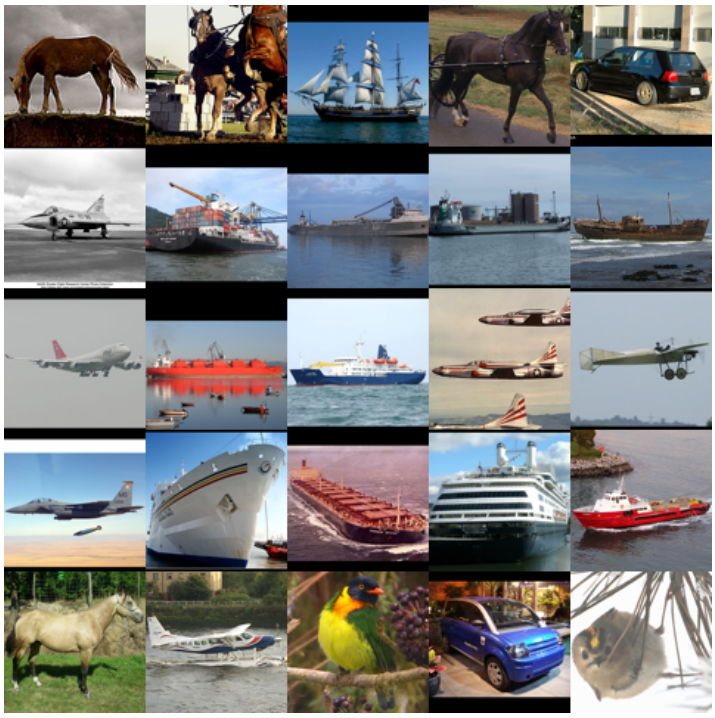


(b) Top 5 least certain

Figure 6: Grayscale dense SIFT with 400 vocabulary size

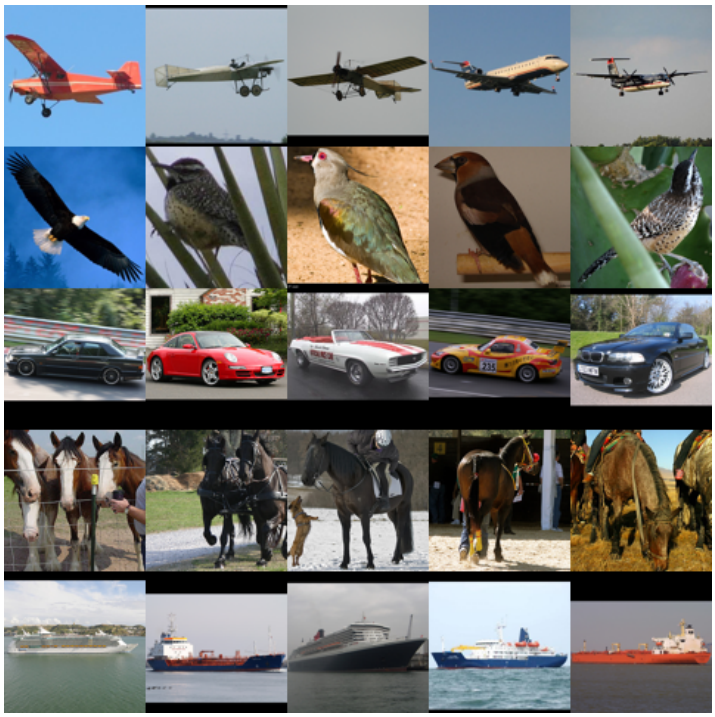


(a) Top 5 most certain

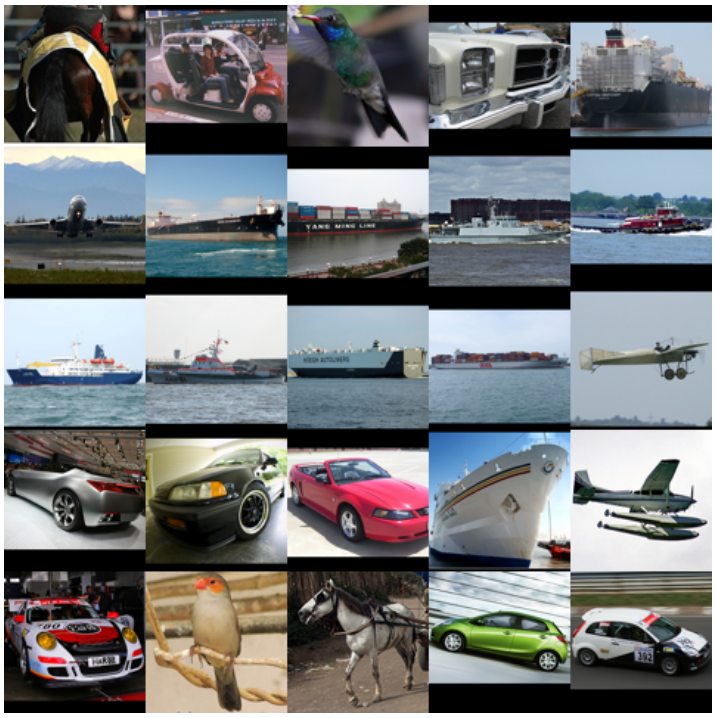


(b) Top 5 least certain

Figure 7: Grayscale dense SIFT with 1000 vocabulary size

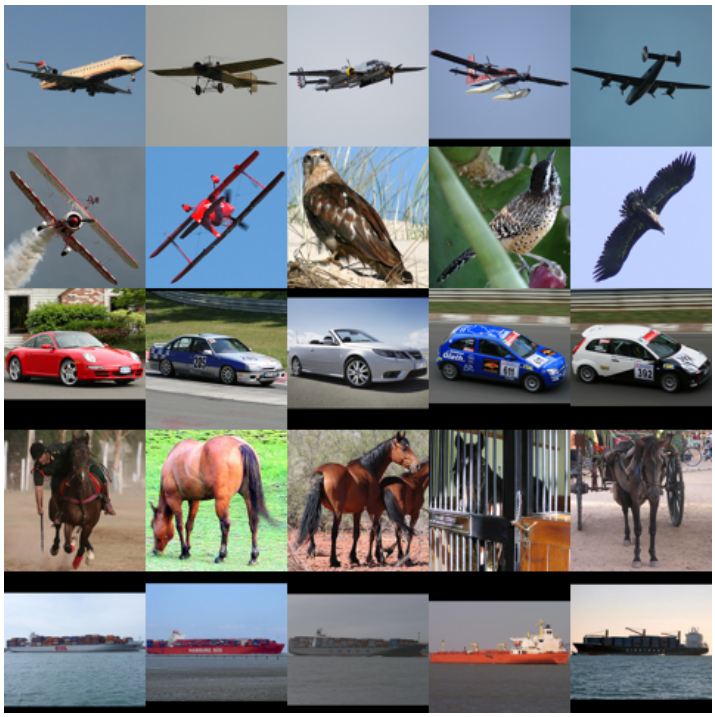


(a) Top 5 most certain

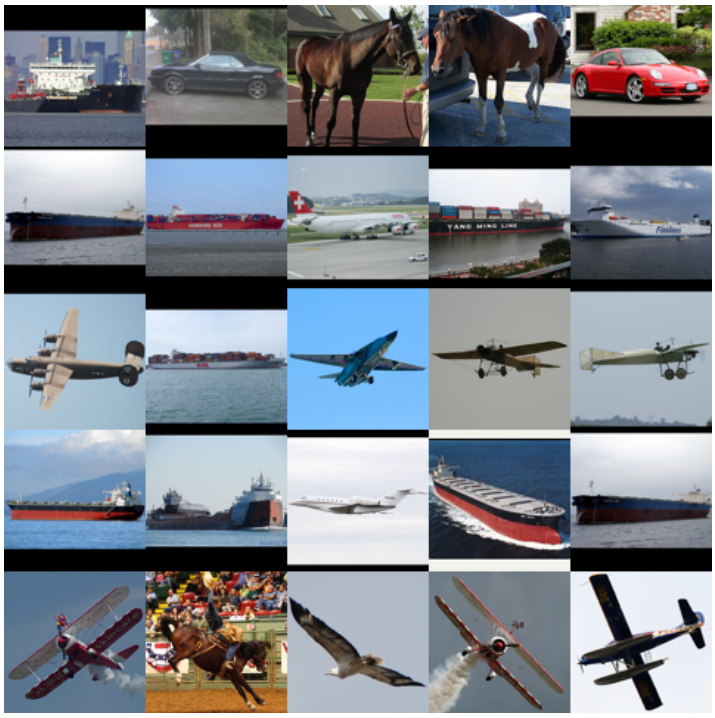


(b) Top 5 least certain

Figure 8: Grayscale dense SIFT with 4000 vocabulary size

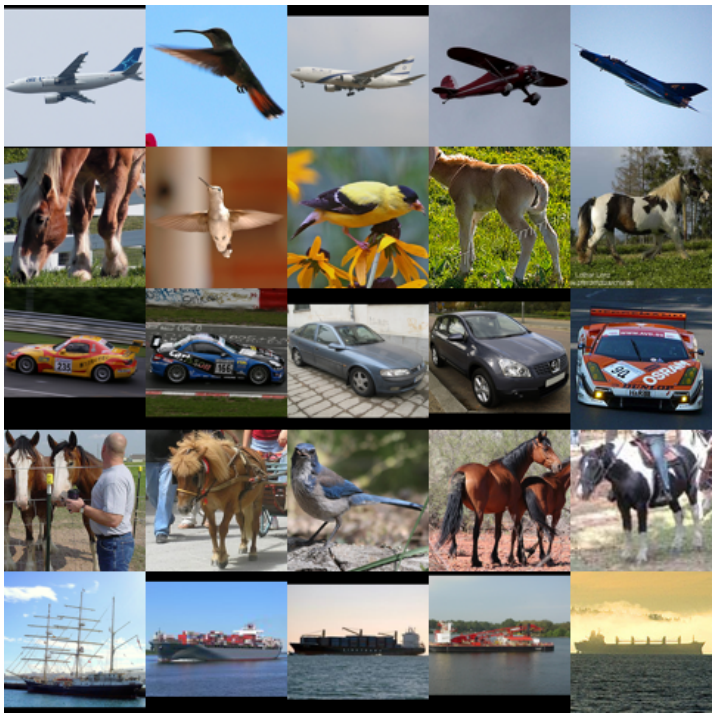


(a) Top 5 most certain

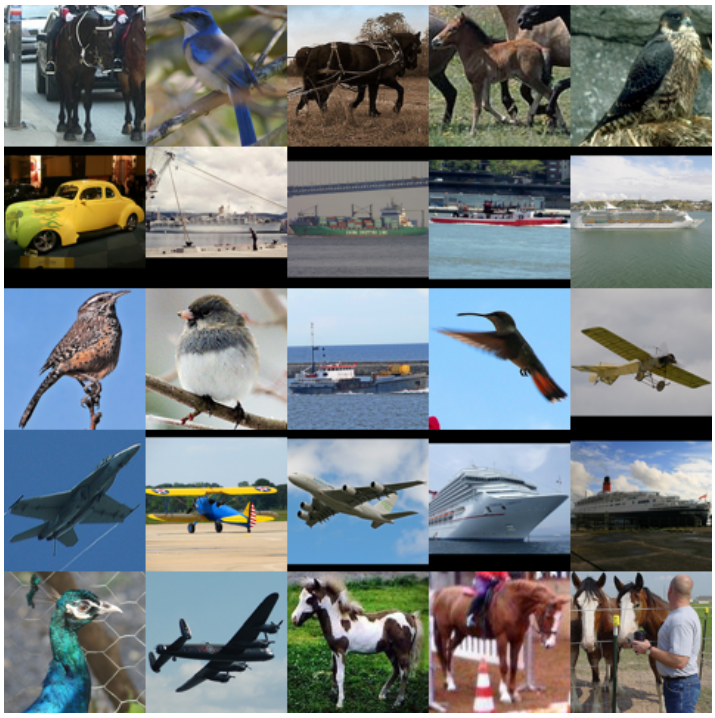


(b) Top 5 least certain

Figure 9: rgbscale regular SIFT with 400 vocabulary size

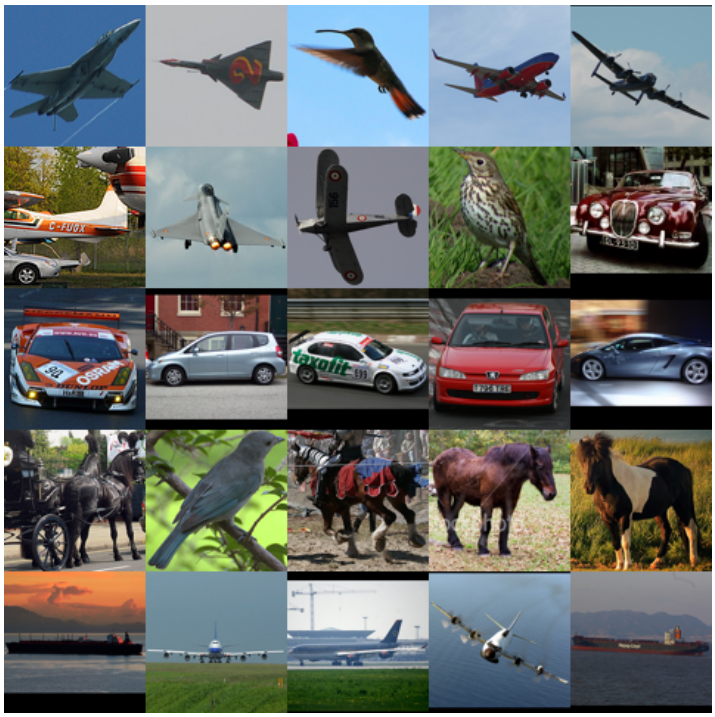


(a) Top 5 most certain

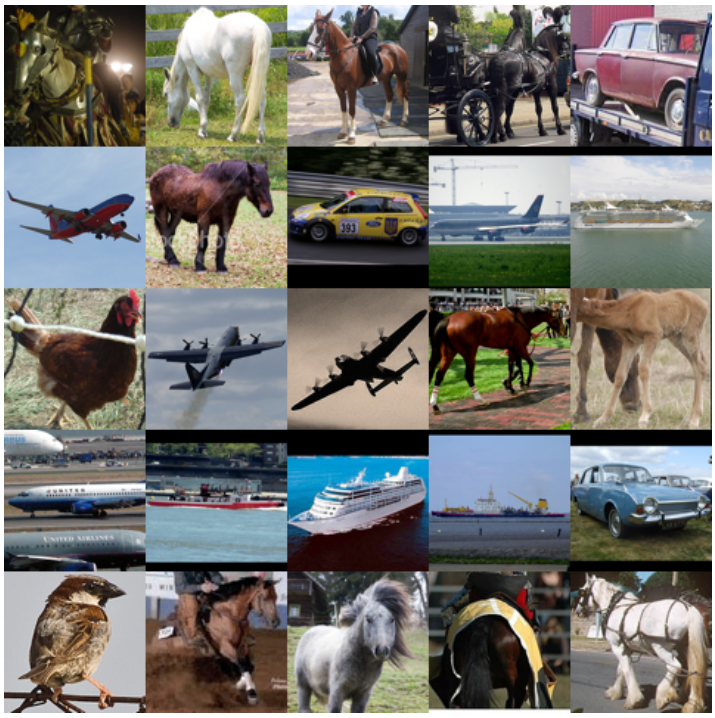


(b) Top 5 least certain

Figure 10: rgbscale regular SIFT with 1000 vocabulary size

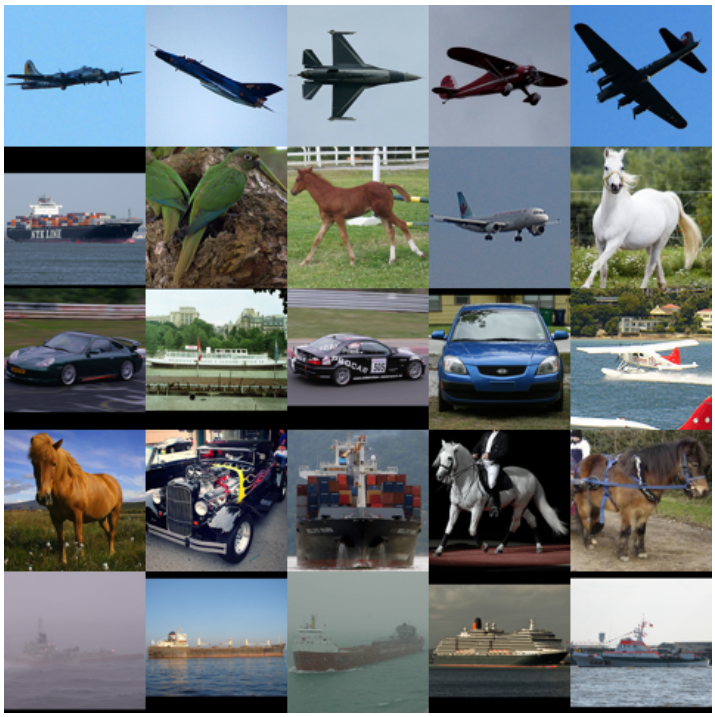


(a) Top 5 most certain

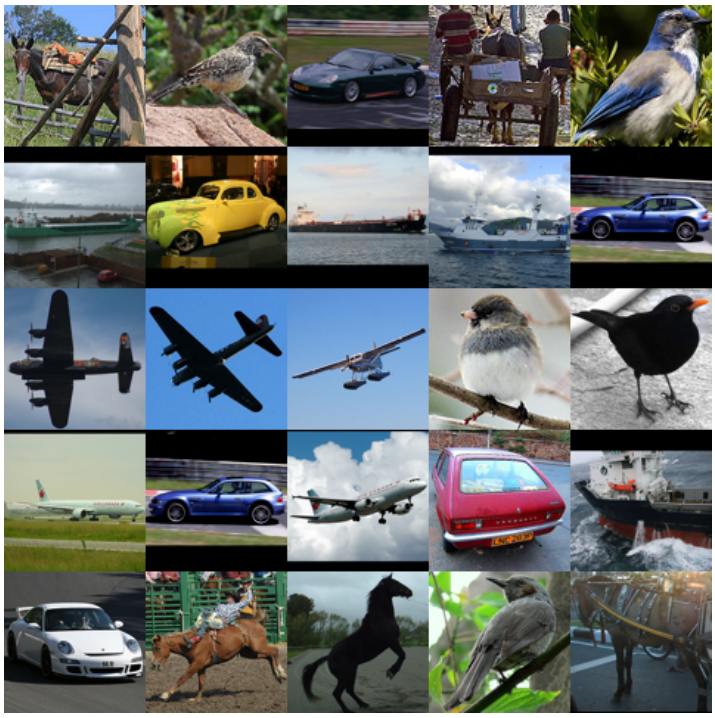


(b) Top 5 least certain

Figure 11: rgbscale regular SIFT with 4000 vocabulary size

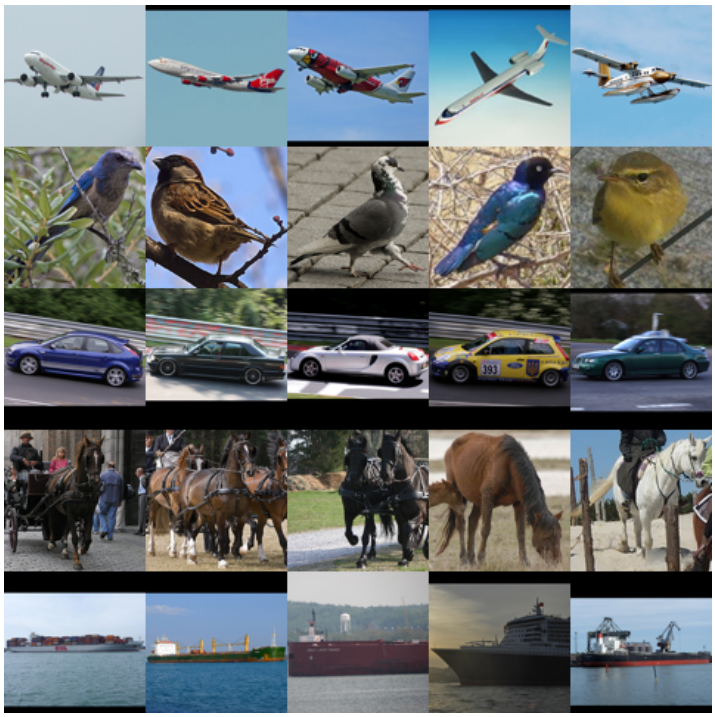


(a) Top 5 most certain

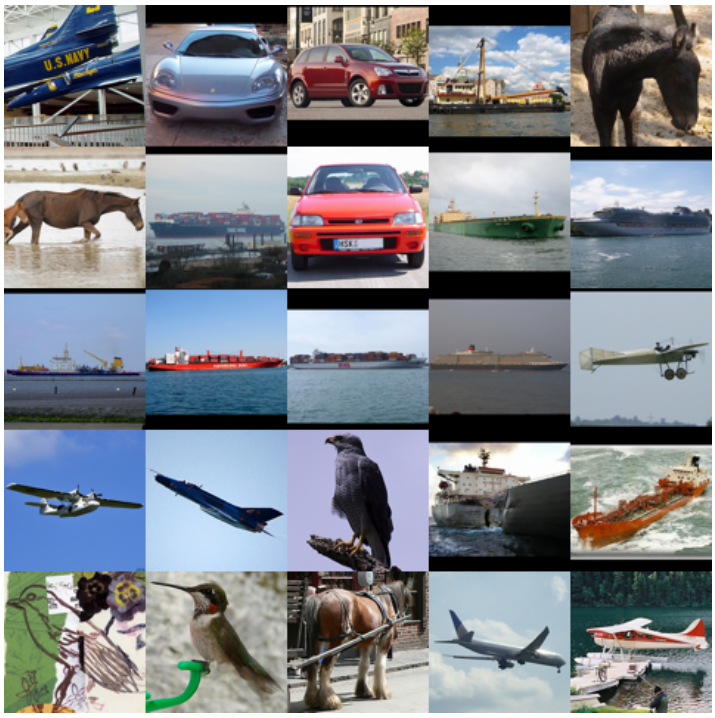


(b) Top 5 least certain

Figure 12: rgbscale dense SIFT with 400 vocabulary size

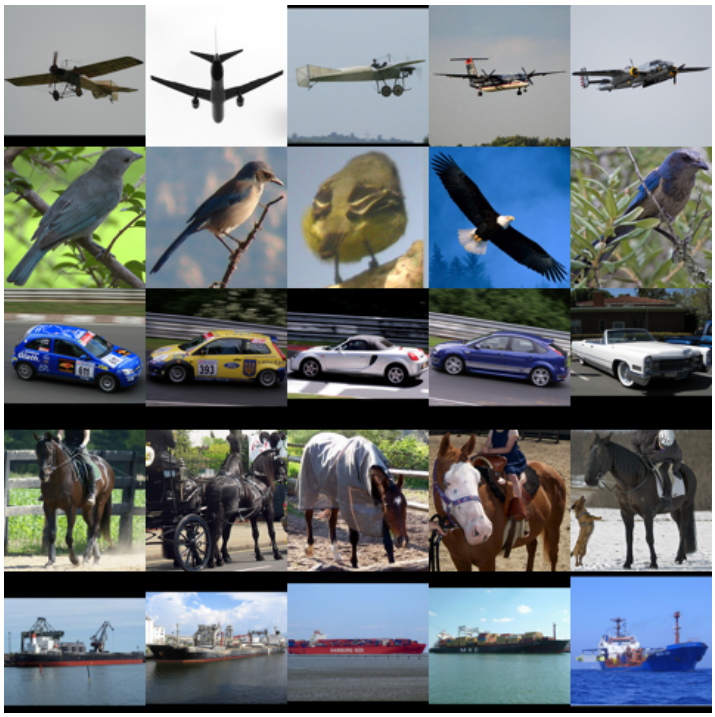


(a) Top 5 most certain

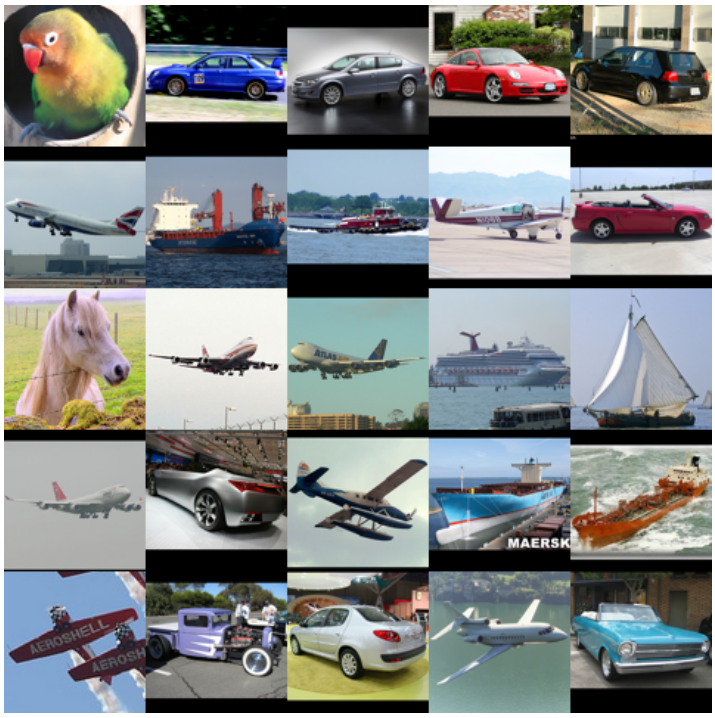


(b) Top 5 least certain

Figure 13: rgbscale dense SIFT with 1000 vocabulary size

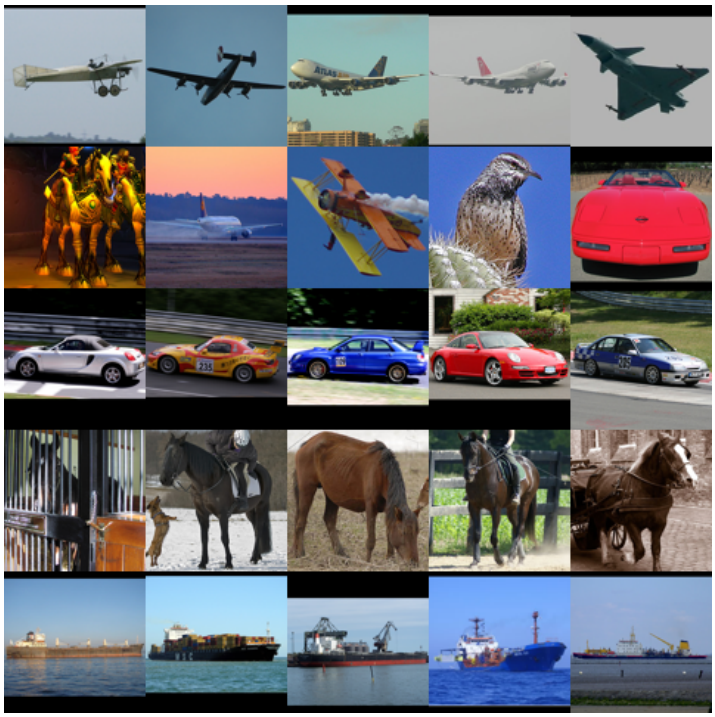


(a) Top 5 most certain

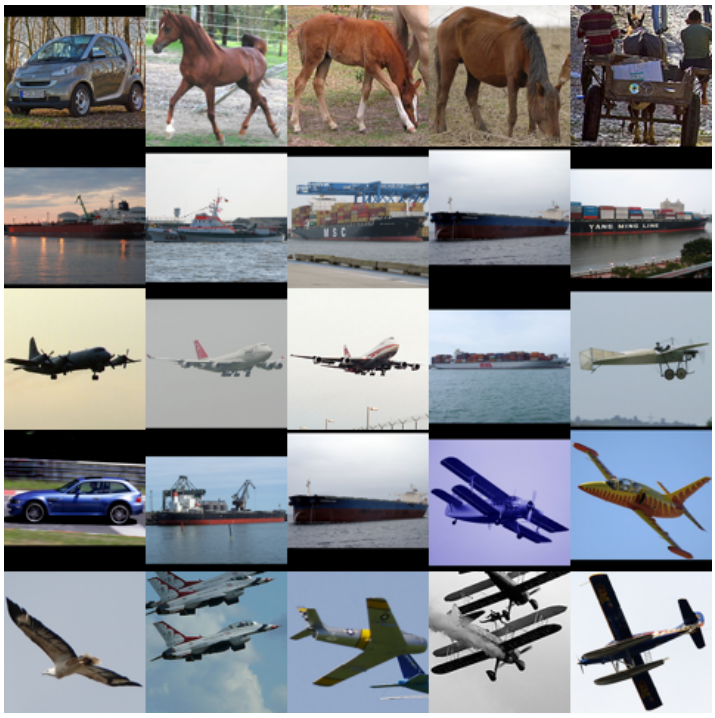


(b) Top 5 least certain

Figure 14: rgbscale dense SIFT with 4000 vocabulary size

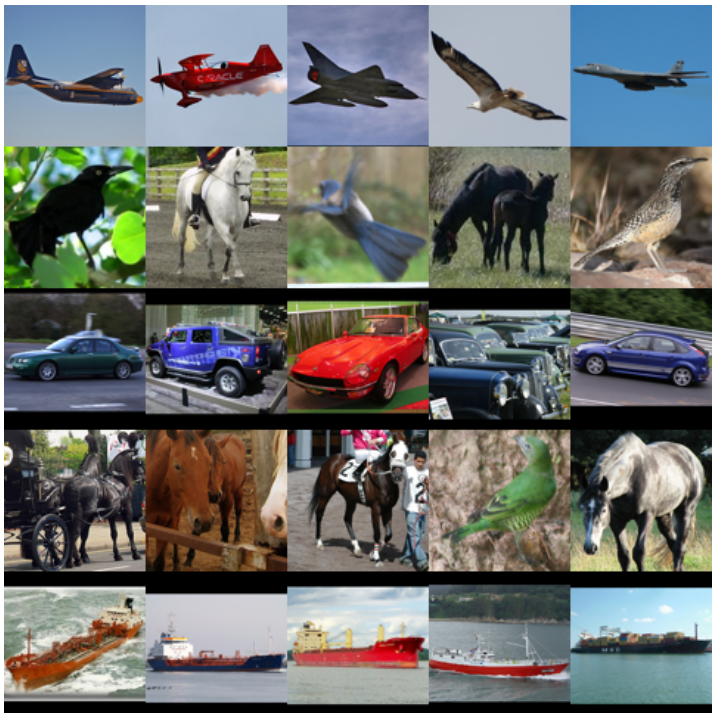


(a) Top 5 most certain

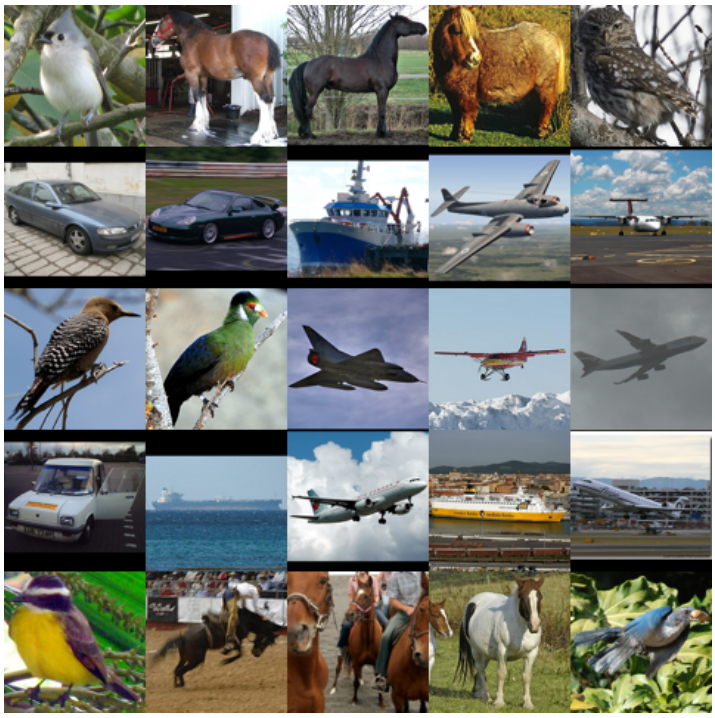


(b) Top 5 least certain

Figure 15: opponentscale regular SIFT with 400 vocabulary size

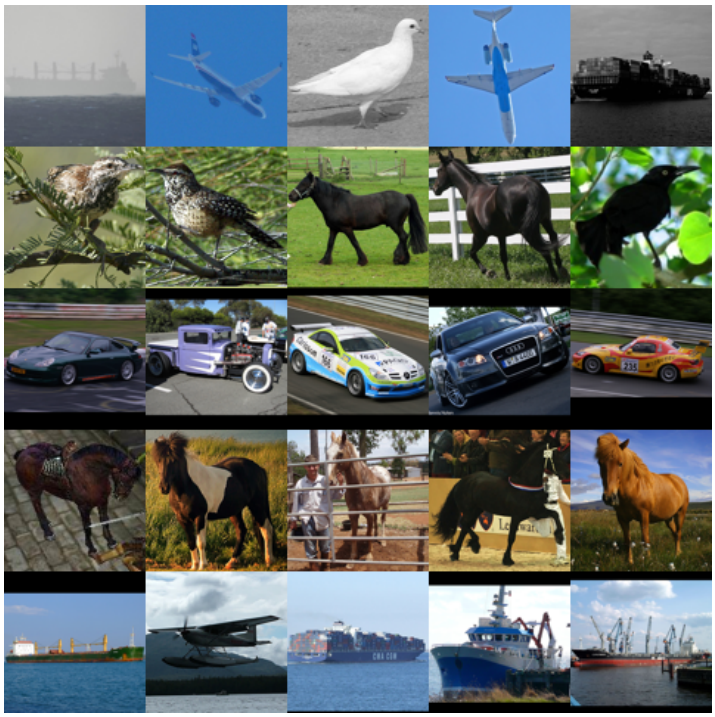


(a) Top 5 most certain



(b) Top 5 least certain

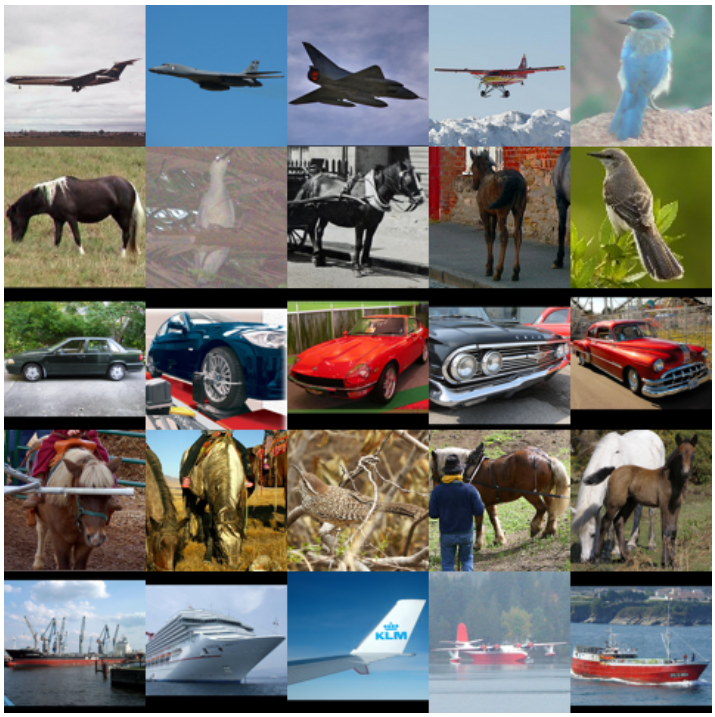
Figure 16: opponentscale regular SIFT with 1000 vocabulary size



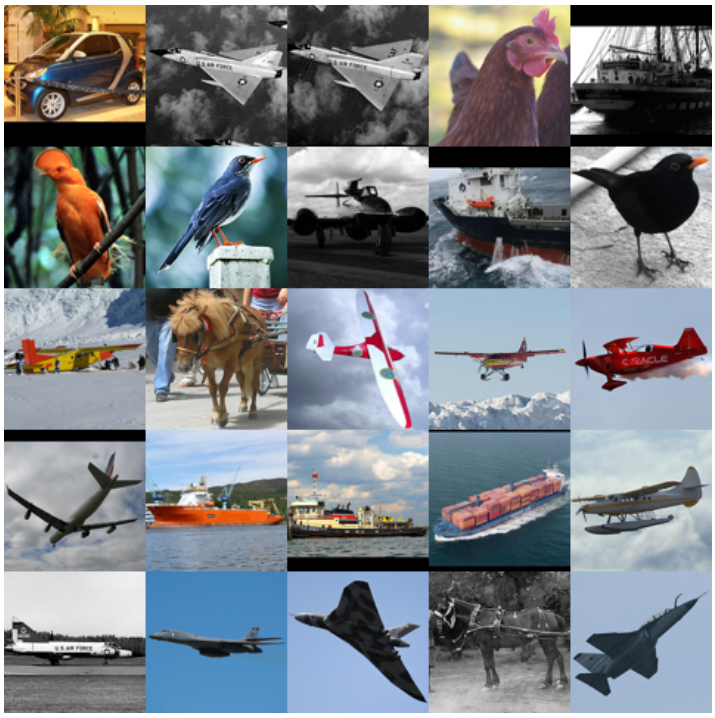
(a) Top 5 most certain

(b) Top 5 least certain

Figure 17: opponentscale regular SIFT with 4000 vocabulary size

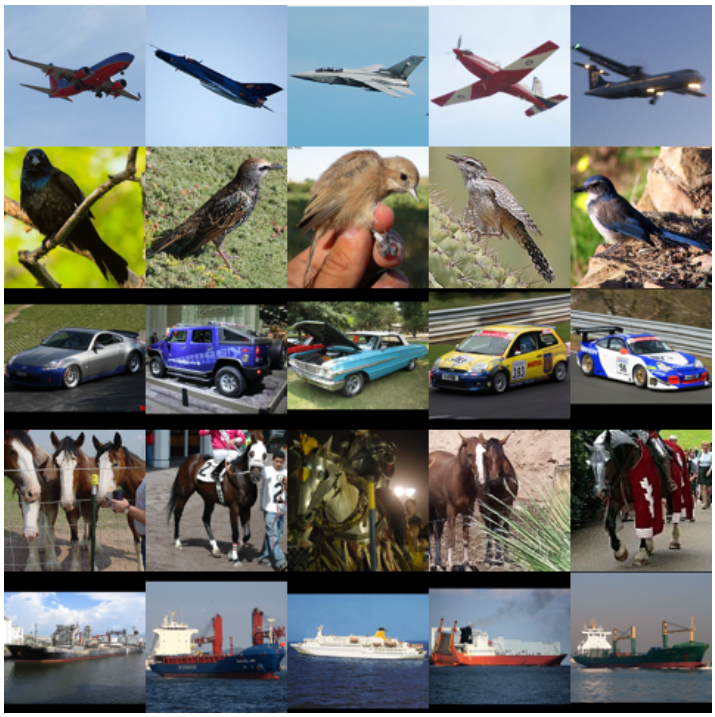


(a) Top 5 most certain

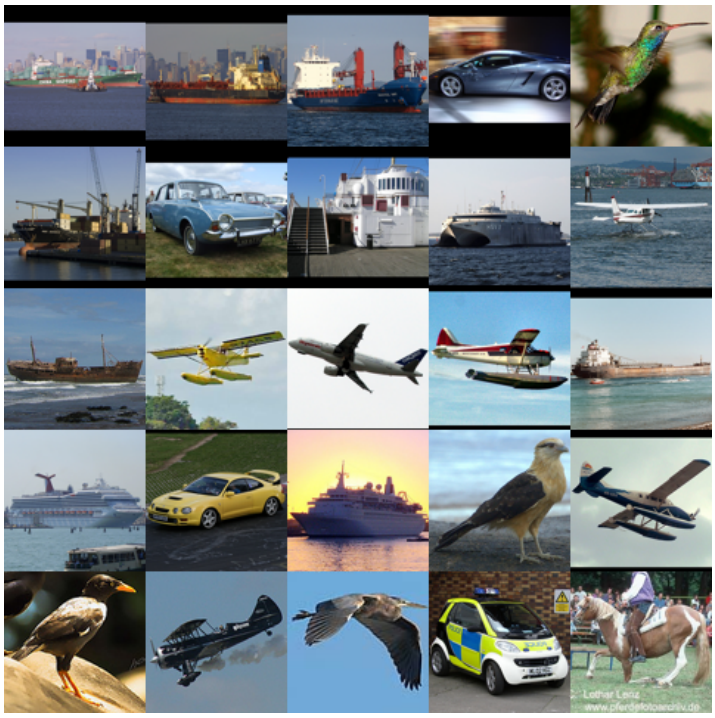


(b) Top 5 least certain

Figure 18: opponentscale dense SIFT with 400 vocabulary size

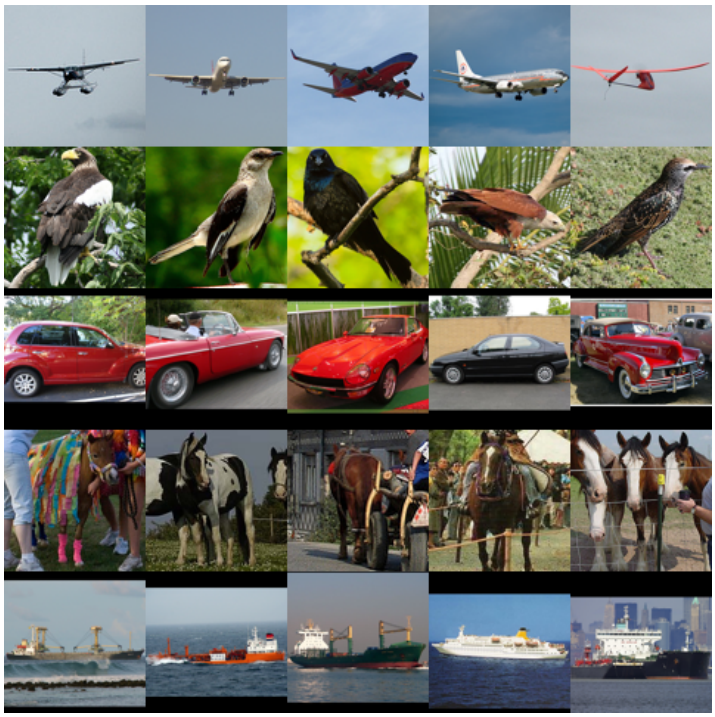


(a) Top 5 most certain



(b) Top 5 least certain

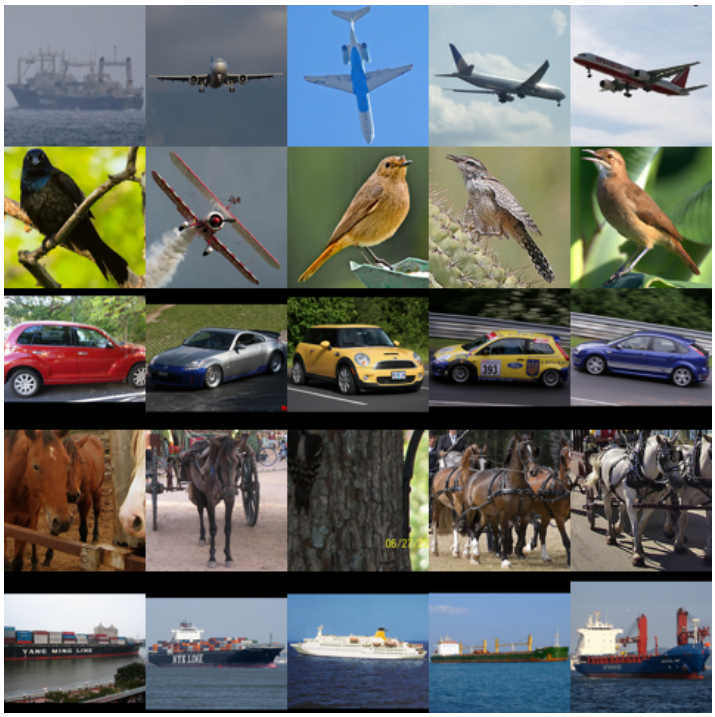
Figure 19: opponentscale dense SIFT with 1000 vocabulary size



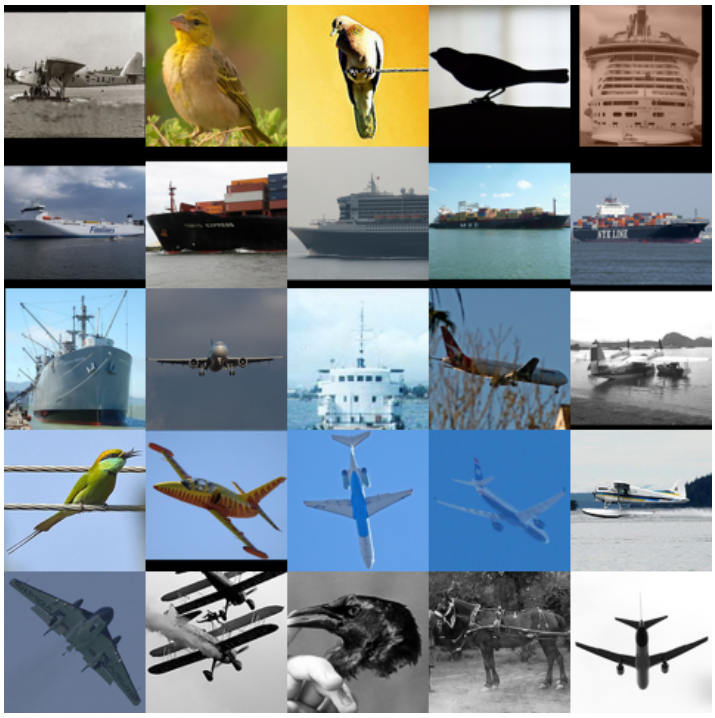
(a) Top 5 most certain

(b) Top 5 least certain

Figure 20: opponentscale dense SIFT with 4000 vocabulary size



(a) Top 5 most certain



(b) Top 5 least certain