Fall 2019

DATA 200: Final Project

Image Classification

***

MOREAU Maël, ZHOU Jingran

Abstract

Building an image classifier is a very common challenge nowadays and it can be used in many different fields. For this project, we built such a classifier in order to predict labels of images from 20 different classes. The feature extraction was the most challenging part, and we finally obtained good prediction results when using Bag of Visual words. Our final classifier which uses a Support Vector Machine reaches an accuracy of around 47% with 5-fold cross validation. The next step would be to use a Convolutional Neural Network to build a more accurate classifier.

Contents

1 Introduction

Our goal in this project is to build an image classifier which should be able to find the right label for a specific image among 20 different labels. We're showing below an example of images we have and we are explaining the different labels right after.

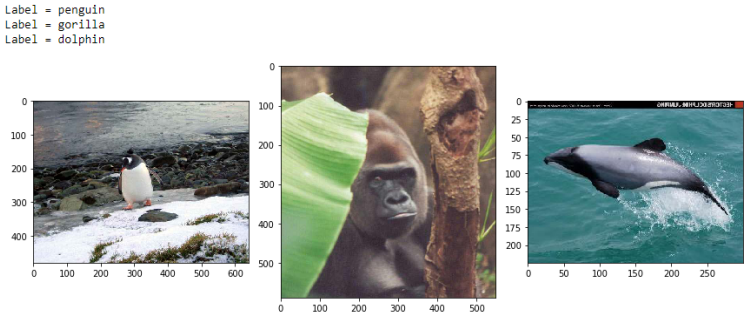

Figure 1: Sample of images

We have 1501 images in the learning set and 716 images in the test set. Those images are broken down into many different categories, such as airplane, crab, blimp, comet, unicorn, etc. We have 20 different labels in total and every image has its own shape and characteristics. In particular, the images have different sizes and aspect ratios, which means our model has to deal with images of varying scales.

With these 1501 images, the first step of preprocessing was to convert them into arrays. However, note that we need to sort the sub-directories (and the images themselves for the "validation" set) to ensure a correct ordering. We will explain in the next part the feature engineering process that we used.

2 Feature engineering

During our EDA, we discovered that not all images are in the RGB format. To be specific, some of them are in gray-scale. In order to unify our dataset, we decided to convert all the images to RGB format before any feature engineering process.

Once we had converted our images into RGB format, we first looked at the distribution of our dataset. We were happy to see that it was generally balanced, except for a few classes such as gorillas.

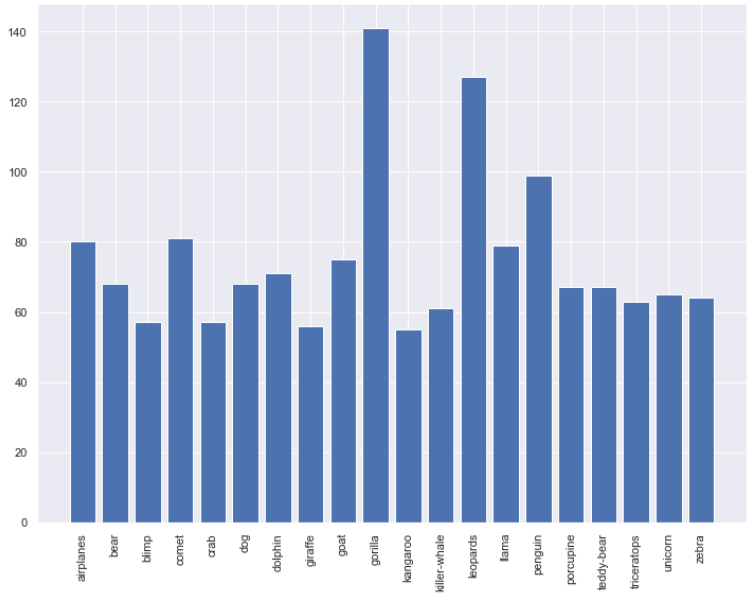

Figure 2: Distribution of our dataset

The most challenging part of this problem was to hand-extract 15 features from the images which could help us to use classic machine learning algorithms to classify them. We first computed simple scalar features in order to get quickly ideas about our dataset. Here's the list of the first scalar features we created:

Size: Returns the pixel size of the image

Aspect_ratio: Returns the aspect ratio of the image

Red_col: Returns the average of the red-channel intensity for the images

Blue_col: Returns the average of the blue-channel intensity for the images

Green_col: Returns the average of the green-channel intensity for the images

Hue: Returns average hue (in HSV)

Saturation: Returns average saturation (in HSV)

Brightness: Returns average brightness value (in HSV)

Frankly speaking, these scalar features are not particularly helpful for prediction because scalar features can only capture *global* or *meta* information about the images. Therefore,

we will not discuss them in great detail. However, they do lead to insightful EDA results. Below, we will discuss two of the interesting features we found.

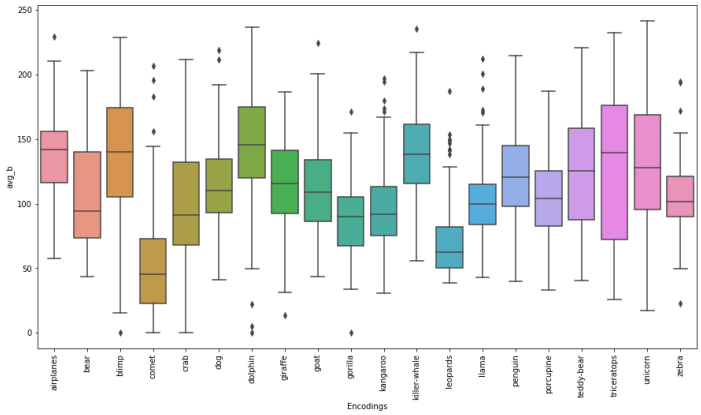

Figure 3: Distribution of average blue channel intensity among 20 classes

First, let us take a look at the average blue channel intensity. It is calculated by averaging the 2D matrix containing only blue intensity values. Notice the comet class has a strikingly low blue intensity. We hypothesize that this is because the background is almost always black for comets, since their pictures are mostly taken in the night. On the contrary, the dolphin class has a very high blue intensity which can be explained by the usual oceanic background on those pictures. In addition, other color-related classes (e.g., red and green) exhibit similar distinctions, but we will not discuss them here due to the page limit.

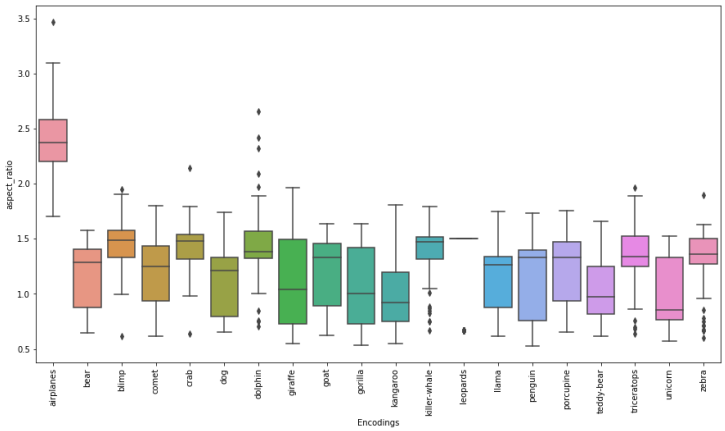

Figure 4: Distribution of aspect ratio among 20 classes

The second interesting feature is aspect ratio. Aspect ratio is simply calculated by dividing width with height. We can see that airplanes have a very particular shape with a low height compared to the width. This is likely due to fact that most airplanes normally fly horizontally, thus their photos are mostly in landscape mode. Also, after examining sample airplane images, we observe that most pictures are taken during take-off or landing, which further supports our reasoning.

So far, we computed only scalar features which were interesting but building a classifier only with these features gave us a very low accuracy (20% for Random Forest algorithm). That is why we tried other features extraction methods which were matrix-based. One of the most common method in computer vision is the bag of visual words methodology. Originally from NLP, this method can be applied to image classification.

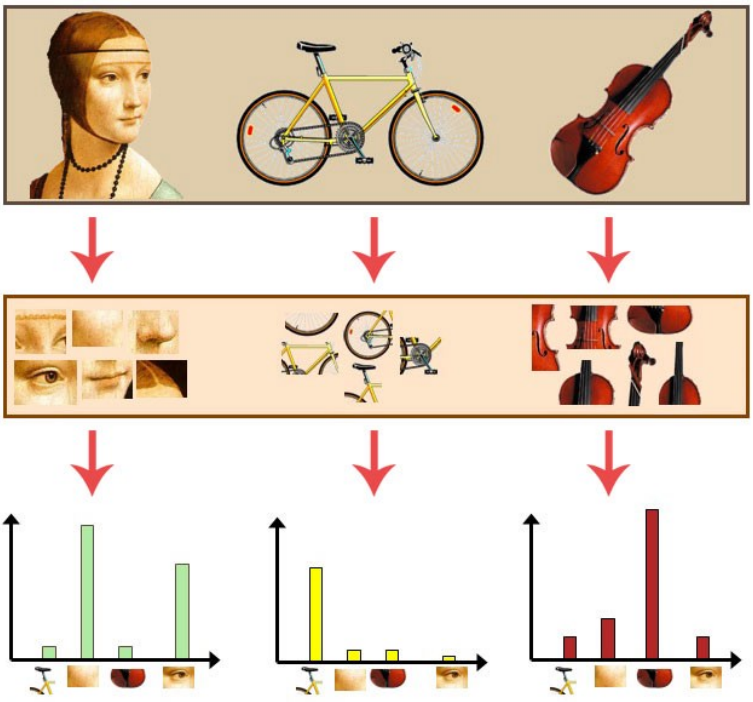

Figure 5: Bag of Visual Words

We use Bag of Visual Words in the following way: First, we generate descriptors for all the images in the training set. Then, we fit a K-Means model on all the descriptors. Now, the K-Means model will be able to categorize each descriptor into one of the k categories. Then, for each image, we can generate a histogram vector describing the categories of that image's descriptors. In this way, we can assign a k-dimensional feature vector to each training image.

We explored three feature extractors, starting with ORB (Oriented FAST and Rotated BRIEF). We expected it to have a good performance since it should theoretically combine the strength of FAST and BRIEF. However, it turned out to be quite ineffective, boosting the validation accuracy by only 1%. We believe ORB's unsatisfactory performance might be due to its inflexibility in adapting to images of varying scales. In the end, we used scale-invariant feature transform (SIFT) and speeded up robust features (SURF), which are the most common extractor for small datasets. We used Mini-Batch K-Means to categorize millions of descriptors instead of the canonical version due to the limits in time and resources, with a batch size of 1,000,000. We trained two K-Means models separately for SIFT and SURF, each with 200 clusters, resulting in 400 new features that increased our accuracy by nearly more than 20%.

Lastly, we also explored another popular feature – histogram of oriented gradients (HOG). However, we did not include it in our final model because of two reasons. Firstly, HOG is mainly used for pedestrian detection, therefore its input image sizes should be uniform, which is not the case for our dataset. Secondly, even with attempts to unify image sizes (e.g., white-patching, cropping, stretching), HOG generates thousands of features for each image, which are too many. However, if we adjust the grid/cell parameters, there would be too few features to describe the outline of the objects accurately. Our validation scores also suggest that using HOG does not yield significant gains. Therefore, we decided not to incorporate this matrix-based feature.

3 Model evaluation

After the previous feature engineering. We started to run several models in order to compute the accuracy. We split the dataset into 75% training data and 15% test data. Then, we performed 5-fold cross validation on the training data for hyper-parameter tuning, and also evaluated our model on the test set. We summarize our results in the next table.

	Logistic Regression	K-Nearest Neighbors	Classification Tree	Random Forest	Support Vector Machine
Validation score	42.69%	21.33%	32.79%	44,24%	**47.56%**
Test score	44.69%	20.35%	33.62%	47.34%	**51.32%**

Table 1: Accuracy of 5-fold cross validation and test

Thanks to the table above, we can see that our best model is obtained with a **SVM** and we got a 5-fold cross validation score of almost **47.56%**. We believe SVM outperforms other models because of the following reasons: First, we are using more than 400 features, and SVM can handle high dimensional data well thanks to the kernel trick. Second, SVM is usually not ideal for categorical variables, while we are only using real-valued features in our prediction. Third, in this particular computer vision project, we are not too concerned with overfitting because of the complex nature of the problem, which enables SVM to achieve good results with relatively simple decision boundaries. Indeed, our SVM model performs the best with the regularization parameter C set to as high as 400.

For an even higher accuracy, we could use the industry standard – Neural Networks, which can even save us the trouble of handpicking features. However, we did not have enough time for Neural Networks in this project. If this were a real-world project, another idea to improve the prediction accuracy would be to gather more images to form a larger learning set.

4 Discussion

In this project, we have made a few design decisions that might have room for improvement. Firstly, we stored both the training and test data frame in as "pickles", which preserves the original type of the data but makes the file size surprisingly large (800 MB for training data). We did not choose the CSV format, which is more memory-efficient, because it stores images as strings, introducing extra data serialization effort. There could perhaps be a better storage format. Secondly, since we chose SVM as our ultimate model, we need to provide an additional scaler to resize all input data, which might be inconvenient to use. Data scaling is needed because SVM runs too slow with unscaled data. In the real world, it may be better

to group the scaler and the SVM together as a new class. Thirdly, when computing the Bag of Visual Words feature, we did not use the full K-Means clustering algorithm since we are running the models locally. If we had more computational resources, we would revert back to the regular K-Means, which will generate more accurate histogram feature vectors. Lastly, although we tested the five models above *individually*, we do not have time to group them into an ensemble model, which could provide better results. In particular, Logistic Regression and Random Forest also seem promising as alternatives to SVM.

5 Conclusion

This computer vision project was both interesting and practical. We were extremely free in terms of feature selection. It was a very long back-and-forth methodology where we tried a variety of features and combinations of hyper-parameters and evaluated our choices using cross validation. In the end, we came up with an SVM classifier with 47.56% of validation accuracy. Although it is not as good as the state-of-the-art Neural Networks models, it is relatively of high accuracy for a traditional machine learning model. The Bag of Visual Words method was particularly helpful for improving the prediction results. Meanwhile, we were quite surprised that ORB failed to outperform SIFT, since ORB should have been a modern improvement to SIFT that combines the strengths of FAST and BRIEF. The most significant limitation of our work is that we did not have time to try Neural Network, which would have very likely been our best image classifier. Therefore, we propose Convolutional Neural Network (CNN) as the next step for this project.