FDS - Fundamentals of Data Science
Final Project

Instructor: Fabio Galasso

Marco Muscas, Robin Münk
Aizirek Doronbekova, Sowmya Kommineni Lokanathbabu

Winter semester - 2021

Abstract

Environment recognition seems like a fun problem, since it allows us to make us of much of the content we have seen during this course. In this project we will use different techniques to go from a set of labelled images to a classifier capable of recognizing the environments shown in the images themselves.

1 Introduction

In this final project we decided to tackle the problem of environment recognition through images. We took inspiration from a work from Oliva and Torralba [1] who provide the dataset and proposed a first pipeline. They used a special technique for identifying features in a scene that they called *Spatial Envelope*.
It consists of an encoding of visual features that allows for a lower dimensionality of the features necessary to describe the image, and a nice interpretability of the features themselves (in our humble opinion, of course).

In our final project we decided to compare their approach to the more traditional techniques, posing the question if their sophisticated method is even necessary, using the tools we familiarized ourselves with during this course.

2 Proposed work

We propose the following pipeline to process images and classify the depicted environments.

- Feature Extraction: using SIFT, ORB and histograms on different color spaces.
- Feature Mapping: using KMeans, to have a consistent number of features for each image - we'll later see why that is necessary.
- Classification: applying different models we have learnt about (Logistic Regression, SVM and Ridge Regression).

We of course included an additional step which is the evaluation of our model. More on that in Section 5.

3 Dataset

The dataset we used is directly taken from the work of Oliva and Torralba [1] available here, which consists of a set of 2600 images that are split into 8 different classes, namely *"Tall Building"*, *"Street"*, *"Highway"*, *"Coast"*, *"Open Country"*, *"Mountain"*, *"Forest"*.

We used a random split of 80% for training, 20% for testing (to be used for the evaluation of our pipeline). The classes were not evenly split, so we made sure to do that in our training dataset.

4 Pipeline

In this section we will explain step-by-step how our work is structured. A summary can be found in Section 2.

4.1 Feature Extraction

Each image is 256x256 pixels large, so we really need a way to reducing the dimensionality of our data, while keeping more "useful" features. As stated before, we will use different techniques for doing so, namely SIFT feature extraction, ORB features extractions, and histogram calculation. All of this was also made intuitive and easy to use thanks to the OpenCV library, more specifically the version installable through PIP found here that includes those algorithms that are not patented, unlike SURF. SIFT was also patented, but the patent expired in March 2020.

4.1.1 SIFT: Scale Invariant Feature Transform

The starting point for learning about SIFT would be the paper from Lowe (2004)[2]. Although meant more to be used in image matching, due to its resistance to scale and rotation of the images, it is also used in the literature as a stepping stone towards image classification in various problem domains. This algorithm gets us the keypoints (localization of the features) and descriptors (gradients histogram for each image patch around a keypoint).

4.1.2 ORB: Another feature extraction method

Introduced by Rublee et al. in 2012 [3], it proposes an alternative method to algorithms like SIFT for feature extraction, which is supposed to be faster than SIFT. The original paper also claims ORB to also bring scale and rotation invariance. That is because they compute the keypoints using FAST from a scale pyramid of a given image.

4.1.3 RGB & HSV histograms

Basically counting the intensities of pixels in the image for a given color space (in our case we considered RGB and HSV) and reporting it all in a histogram of different number of bins. During the first assignment we have seen that, at least for this family of problems, the histograms are not a good candidate for feature extraction, as the color information is hardly representative of an image, and any representation of arguably useful features inside an image (the peak of a mountain for example?) is simply non-existent.

SIFT and ORB on the other hand do not even "care" about a color space, as the features are extracted using just a grayscale image.

We quickly discarded RGB and HSV histograms in the feature extraction part, so we will exclude it from our analysis.

4.2 Feature Mapping

Let's recap a bit: the number of features found inside an image is highly variable, therefore we suppose each image to have a different number of descriptors from the rest. So how can we standardize the dimension of the feature maps? **Clustering**.

The algorithm that we choose to use for feature extraction, luckily, gets us vectors of a consistent number of dimension: SIFT returns descriptors in $\mathbb{N}^{128}$, while ORB in $\mathbb{N}^{32}$.

The next step would be to get a Bag of Visual Words (BoVW), once we do that we're ready for the classification!

4.3 Getting the Bag of Visual Words through KMeans

As stated in the last presentation of our project: a bag of visual words is a one-hot vector indicating the presence of visual "terms" (features).

In order to do so, we will first need a desired number of features for our feature map (BoVW). We have simply to specify a number of desired elements for our BoVW and train a clustering model specifying the number of features as the number of desired clusters.

For clustering we chose to go with the famous KMeans, more specifically the **minibatch** KMeans implementation. That is justified by the fact that KMeans would require to store the distance matrix of *all* descriptors

from *all* the clusters in memory. For us, students with laptops that easily give up, it was simply not feasible since we would get more than a million descriptors in total and possibly hundreds or thousands of clusters.

Therefore the minibatch variant was ideal, after expecting the results in the clustering to be slightly worse than the original KMeans counterpart.

5 Classification

We decided to choose three models to perform our classification on the feature maps, namely Logistic Regression, Ridge Regression and SVM. As per our last presentation, we also wanted to test additional models and hyperparameters, plus some method that could be fun! So, we added cross validation methods, which are supposed to help with overfitting, and also some dropout methods to see if it changes anything.

About the dropout, we tried different versions. One would try and drop elements in the feature maps according to a probability related to the number of occurrence of each feature:

For a sample $X_i \in \mathbb{N}^d$, where d is the number of features (size of the bag of visual words), each element $X_{i_j} \in \mathbb{N}$ had a probability of being dropped of

$$\mathbb{P}(\text{drop}_j(X_i)) = 1 - \frac{X_{i_j}}{\sum_j X_{i_j}}$$

To simply interpret it, the more a feature was encountered, the less likely it was to be dropped. Sampling from a Bernoulli k times, we would get another vector $\in [0, k]^d$, then transformed in $[0, 1]^d$, so that when multiplied element-wise by X_i would "drop" some of the elements.

That did not work well though! Another version, arguably simpler, requires us to set an arbitrary probability p' for an element to be dropped. Then after sampling a vector in $[0, 1]^d$ from a Bernoulli $v = [0, 1, 0, 0, ...]^d \sim$ Bernoulli$(p = p')$ would be transformed into $\bar{1} - v$ using the logical *not*. That is because the sampled vector by itself indicates us what to drop (the 1's), but a multplication would drop everything else, that's why we transform it and perform an elementwise multiplication again.

That worked a bit better, even got a 70% accuracy, but no consistent results.

6 Experiments

In this chapter we present the setup and main results of the experiments we conducted to accompany our theoretical ideas.

6.1 Hyperparameter Search

In order to get the best results of our proposed pipeline we conducted an extensive hyperparameter search. Here fine-tuned the parameters that are specific to our pipeline, not the classification models. For this we implemented our own grid search completely from scratch! This allowed us to adjust the search specifically to what we needed. The variables defining our pipeline are listed below, the numbers in brackets indicate all values that were assumed during the grid search.

- The *feature extraction model* [ORB, SIFT, RGB&HSV]. As thoroughly explained in Section 4.1 we tried a large variety of different models to extract the image features.
- The *batch size* [128, 256, 512] of the minibatch Kmeans.
- The *BoVW size* [128, 256, 1024, 2048] which is the number of clusters that we let out Kmeans classify the descriptors into. At the same time this is the size of the input to our classification model. Its size is of crucial importance to the pipeline, as it directly influences how much the information is condensed into. Thus justifying the large set of numbers, spanning multiple orders of magnitude.
- The *dropout rate* [0.1, 0.2, 0.3, 0.4], to be used when dropout is enabled in the feature mapper.
- The *classification model* [SVM, LogReg] used to perform the final classification. Sidenote: We also tried the linear classificaion model Ridge regression model, but it did not perform nearly as well as the other models.

6.2 Results

Let's now see the best results! We will not consider them all as we have over 180, but the following table gives an overview of the pipeline configurations which achieved the highest accuracies in our experiments.

Batch size	BoVW Size	Classification Model	Dropout	Dropout rate	Accuracy
512	2048	SVM	False	-	0.71
256	512	SVM	False	-	0.71
256	1024	SVM	False	-	0.71
512	2048	CV Log Reg	False	-	0.70
512	2048	CV Log Reg	True	0.4	0.70
512	2048	Log Reg	False	-	0.70
256	1024	SVM	False	-	0.70
512	2048	Log Reg	True	0.1	0.70
512	2048	CV Log Reg	True	0.1	0.69
512	1024	CV Log Reg	False	-	0.69
512	2048	CV Log Reg	True	0.2	0.69

Table 1: Overview of the pipeline configurations with the highest accuracy

As a side note, to be able to fit the table, I had to shave off a few columns, which luckily had all consistent values for the feature extraction method and mapping. We used SIFT and minibatch-KMeans in each run.

7 Conclusion

In conclusion, we can say this was a fun project! The results we have are satisfactory, considering that we know about that certain classes could be aggregated together (e.g. *street* and *highway*, *insidecity* and *tallbuilding*) - though we chose not to do that as it almost sounds like cheating, we liked the original dataset and problem with no alterations.

The best results we can see were around 71%, after that there seems to be a ceiling we cannot get through with these methods. Maybe it requires something a bit more modern, CNNs perhaps? However in general we are very content with how far we got using the traditional methods only. Thank you for your attention!

References

[1] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vis.*, 42(3):145–175, 2001.

[2] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, 2004.

[3] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. ORB: an efficient alternative to SIFT or SURF. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc Van Gool, editors, *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 2564–2571. IEEE Computer Society, 2011.