Homework 2

RInS

Authors: Drejc Pesjak, David Trafela, Urban Plaskan

Date: 5.5.2021

Table of Contents

1. Instructions (copied)....2
2. Dataset generation/retrieval....2
3. Color classifiers....2
  3.1 kNN classifier by Ahmet Özlü....2
  3.2 Hard-coded color range detector....2
  3.3 Comparison of histograms....3
  3.4 Convolutional neural network....4
  3.5 kNN classifier....4
4. Evaluation....4
  4.1 kNN classifier by Ahmet Özlü....5
  4.2 Hard-coded color range detector....5
  4.3 Comparison of histograms....5
  4.4 Convolutional neural network....5
  4.5 kNN classifier....6

1. Instructions (copied)

The second homework from perception is to implement and evaluate a reliable colour recognition method.

The goal is to train a classifier to recognize at least six colours: red, green, blue, yellow, white, black. Find different objects of these colours or paint or print them (in different shades and tones) on the paper. Then take a number of images of these objects under different conditions (illumination, viewing angles etc.) and split them on the training and test sets. Using the training set train a classifier and evaluate its performance on the test set. Evaluate different classifiers, try using different colour spaces, and report the results.

2. Dataset generation/retrieval

The dataset of 2900 images was retrieved from the google images site using the automated method described here: https://www.pyimagesearch.com/2017/12/04/how-to-create-a-deep-learning-dataset-using-google-images/. Search terms like the name of each color were used. According to their color images were separated into different folders at the time of download.

3. Color classifiers

We've experimented with 5 classifiers, of which 4 were implemented by us.

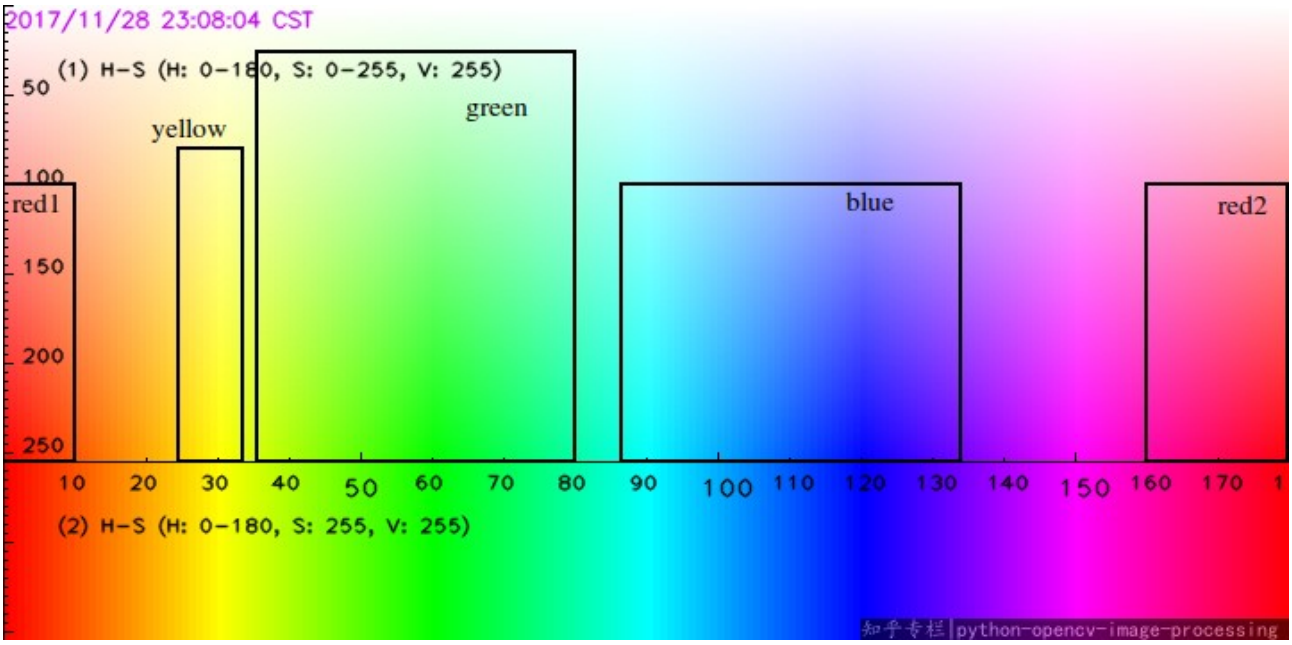
3.1 kNN classifier by `Ahmet Özlü`

He uses the k-Nearest Neighbors method on RGB color histograms by calculating the euclidean distance. The code and more in depth explanation is available at his github (https://github.com/ahmetozlu/color_recognition).

3.2 Hard-coded color range detector

Using predefined color ranges in HSV color space, we take an image, count the number of pixels that fall in each range and predict the color with most "votes".

Ranges are depicted on the following picture:



Pixels are considered black, if V is less then 50.

(Note: as an improvement HSL color space could be used, which allows us an easier and more efficient way of defining ranges. Especially with colors like black and white that can nicely be separated along the Luminance value.)

3.3 Comparison of histograms

At the start this algorithm takes HSV images, makes a histogram, and then builds an average color histogram for each color.

To predict the we compare the histograms for **hue**. The histogram of the image and the averages are compared by calculating Bhattacharyya distance implemented in the cv2 python library, where the histograms are taken as a probability density function.

$$BC(p,q) = \sum_{x \in X} \sqrt{p(x)q(x)}$$

3.4 Convolutional neural network

The neural network was trained on the above specified dataset (2.). Actually the CNN takes as input a standard RGB image of size 128×128, which means most of the images need to be resized. The network uses the Adam optimizer and it has the following structure:

(a few conv. layers with ReLU activation function and max-pooling; at the end two dense layers are added, last one having 6 outputs which is equal to the number of classes)

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 126, 126, 128)     3584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 63, 63, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 61, 61, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 30, 30, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 28, 28, 64)        147520
_________________________________________________________________
flatten_1 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               12845312
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 1542
=================================================================
Total params: 13,293,126
Trainable params: 13,293,126
Non-trainable params: 0
_________________________________________________________________
```

Training was done in 10 epochs.

3.5 kNN classifier

This color detector was implemented by our team. It extracts 3d color histograms (with 8 bins in every dimension) from images encoded in the HSL color space. The histograms are flattend and used as a feature vector.

When predicting kNN finds 5 (default) nearest neighbours and predicts the class with most “votes”.

4. Evaluation

First three classifiers were evaluated on 100 images of each color and attained the following results:

4.1 kNN classifier by Ahmet Özlü

Blue color	Green color	Red color	Yellow color	White	Black
57	57	59	50	97	94

Total accuracy: **69.00%**

4.2 Hard-coded color range detector

Blue color	Green color	Red color	Yellow color	White	Black
85	92	82	83	100	99

Total accuracy: **90.17%**

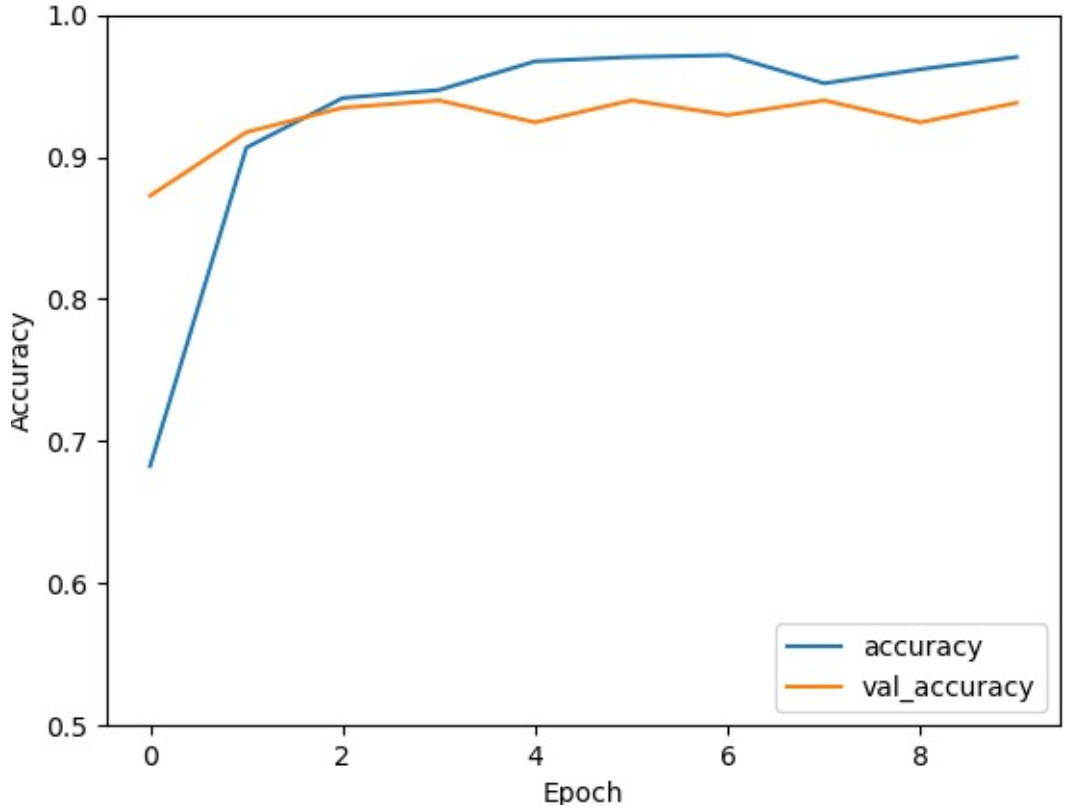
4.3 Comparison of histograms

Blue color	Green color	Red color	Yellow color	White	Black
95	90	92	87	63	35

Total accuracy: **77.00%**

4.4 Convolutional neural network

On the picture are plotted the *accuracy* on the train set and *val_accuracy* on the validation set, calculated during training for each epoch.



Total accuracy: **93.81%**

4.5 kNN classifier

This classifier was evaluated for different k values. With best accuracy of **80.69%** at k=5.

```
knn= 1   -> acc=78.48%
knn= 2   -> acc=76.83%
knn= 3   -> acc=79.59%
knn= 4   -> acc=78.21%
knn= 5   -> acc=80.69%
knn= 6   -> acc=79.03%
knn= 7   -> acc=79.31%
knn= 8   -> acc=79.17%
knn= 9   -> acc=78.48%
knn= 10   -> acc=78.48%
knn= 11   -> acc=78.07%
knn= 12   -> acc=77.93%
knn= 13   -> acc=77.24%
knn= 14   -> acc=76.55%
knn= 15   -> acc=76.41%
knn= 20   -> acc=75.86%
```