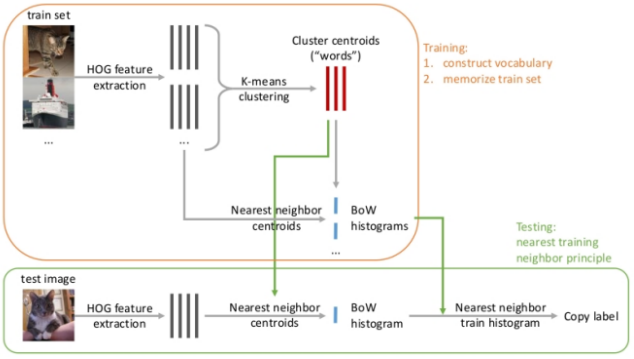
GHILENE RAYANE
LASSO JARAMILLO NICOLAS

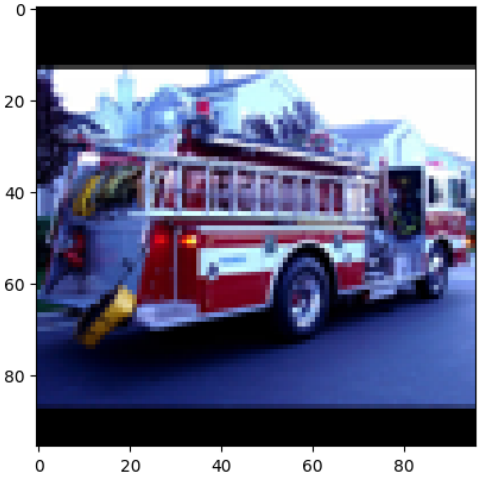
Image Processing
Lab2 Image Classification

The goal of this lab: *is to build an image classier based on the bag-of-words (BoW) approach and the nearest neighbors algorithm. Here's a preview of the pipeline we will follow in our implementation*



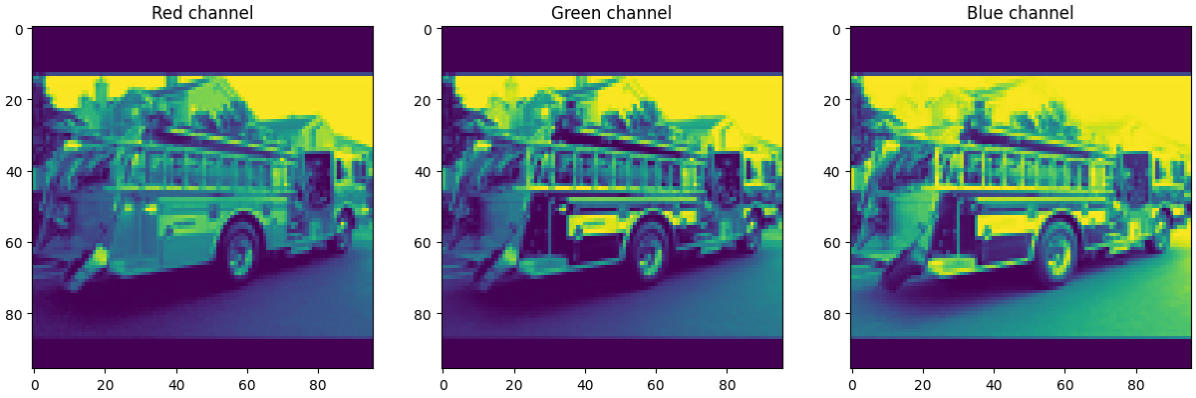
1. Dataset

We started by importing our dataset and performing preprocessing on it.
The dataset contains 96*96 color images distributed over ten classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck.



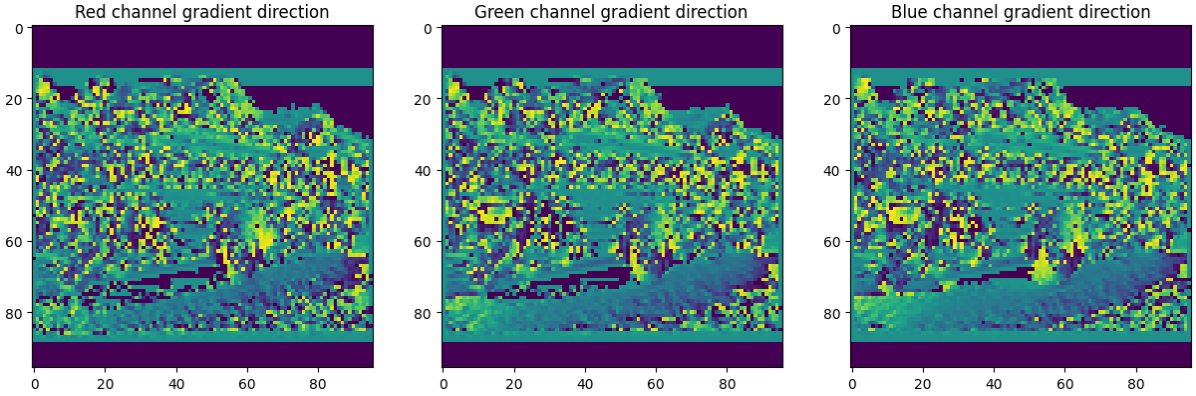
Plot 1: Image from train dataset

We started by separating the three channels of the image (red, blue, and green), we will use these three channels to calculate the hog descriptors and then we will concatenate them to get a more accurate result



Plot 2: Separation of the image channels

Similarly to the last lab session, we used a Sobel filter on our images for feature extraction and we calculated the gradient magnitude and direction, here's the result:



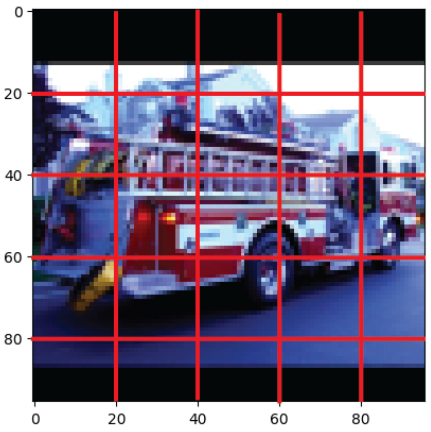
Plot 3: Gradient direction of the three channels

Feature Description with Histograms of Oriented Gradients (HOG):

Step 1: We started by defining the function "grid" that returns for each image the column (x) indices and the corresponding row (y), this function will be used to access different cells of our image.

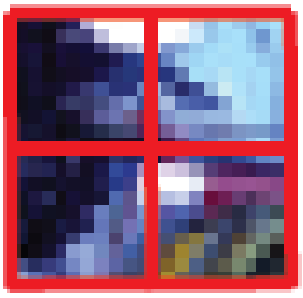
Step 2: We defined the function get_Cells to access individual cells and calculate the different histograms using the get_histograms, which we then apply to the entire image using the function get_all_histograms.

First, we divided the image into 6*6 cells, and then we calculated the histogram of the oriented gradients for each image



*Plot 4: 6*6 Cells*

And then each cell is divided into 4 mini-cells:



Then the mini-cell is separated into 4 cells which gives us 36 center points in our image

We apply the get_all_histograms to the three channels of each image and then concatenate the output of these three channels to get the global HOG response.

Visual Vocabulary Construction with K-means Clustering

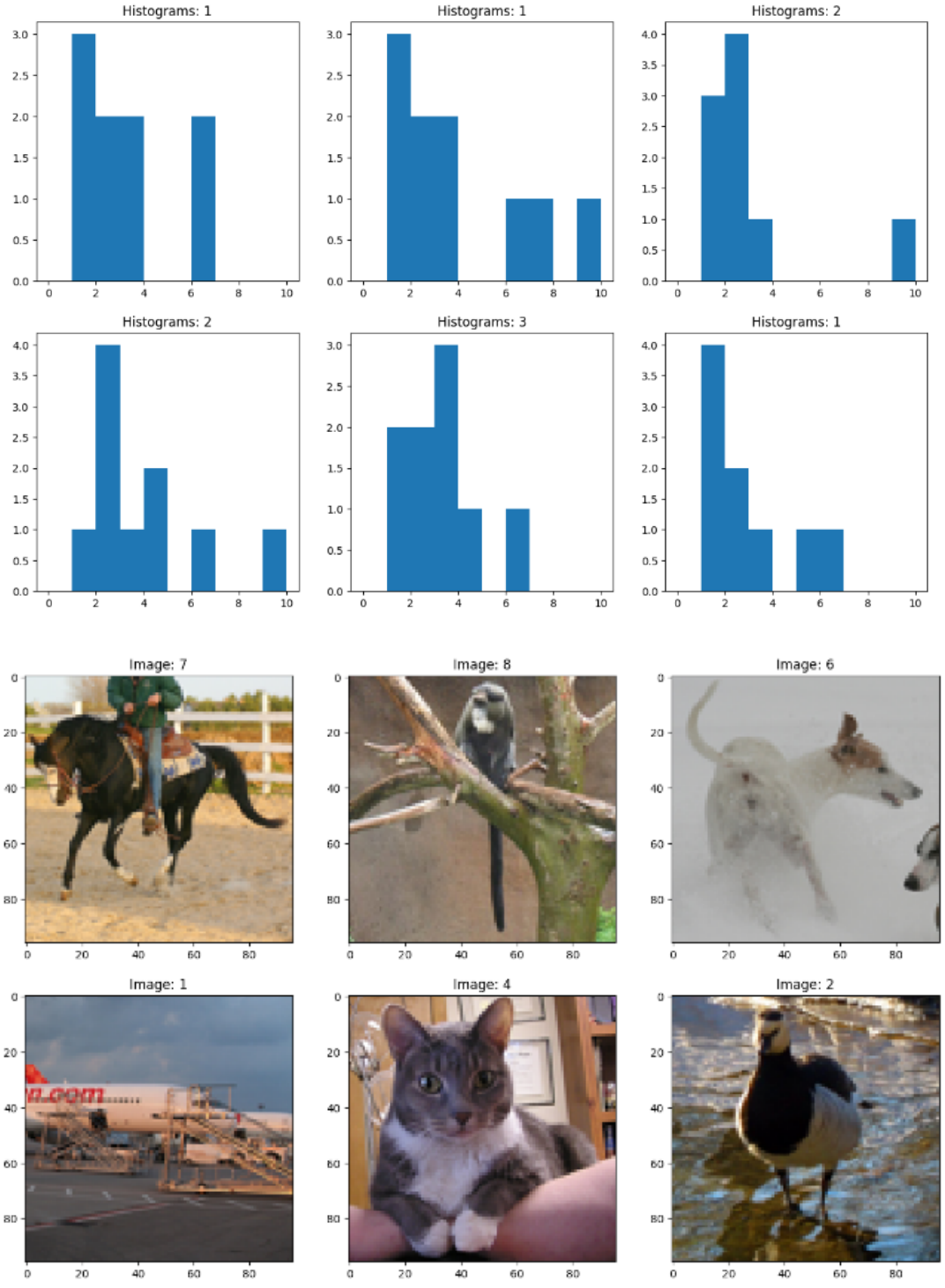
We write a simple implementation of the k means clustering algorithm where our data points, consisting of the concatenated histograms of the HOG features, are represented in a 128-D space.

```
def kmeans(X, k, max_iter=100):

    centroids = np.random.permutation(X)[:k]
    clusters = []

    #For each iteration

    for i in range(max_iter):
        distances = cdist(X, centroids)
        clusters = np.argmin(distances, axis=1)
        centroids = np.array([X[clusters == i].mean(axis=0) for i in
range(k)])

    #Return the centroids and the clusters
    return centroids, clusters
```

We are going to use this for the BoW histogram representation.

BoW Histogram Representation

For the BoW representation, we want to see the number of HOG descriptors that are close to each one of the defined clusters. We use our k-means to get this histogram representation, and then we develop a nearest neighbor algorithm, to compute the L2 distance to all the images in the dataset.



Plot 5: BoW representation of test images

10 Class Classification Problem

Even after doing all the right calculations during this practice, we arrived at a classifier that performed randomly over the STL-10 Dataset, and with all the 10 different classes.

With an accuracy of 12% we can conclude that the classifier performs randomly. With fewer classes, the accuracy would increase, cause we have less variance- Sadly, because of the time constraint, we could perform more experiments to show this.

end