Assignment 1

Sangeeta Yadav (15318)

February 6, 2022

1 Question 1

Image 1 and Image 2 as shown below, are considered for keypoint extraction.

1.1 Steps of scale-space extrema detection

Let say the given image is I. The DoG image is represented as $D(x, y, \sigma)$ and can be computed from the difference of two nearby scaled images separated by a multiplicative factor k.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{1}$$

The convolved images are grouped by octave. An octave corresponds to doubling the value of σ , and the value of k is selected so that a fixed number of blurred images are generated per octave. This also ensures the same number of DoG images are obtained per octave. Keypoints are identified as local maxima or minima of the DoG images across scales. Each pixel in a DoG image is compared to its 8 neighbors at the same scale, and the 9 corresponding neighbors at neighboring scales. If the pixel is a local maximum or minimum, it is selected as a candidate keypoint.

1.2 Implementation details

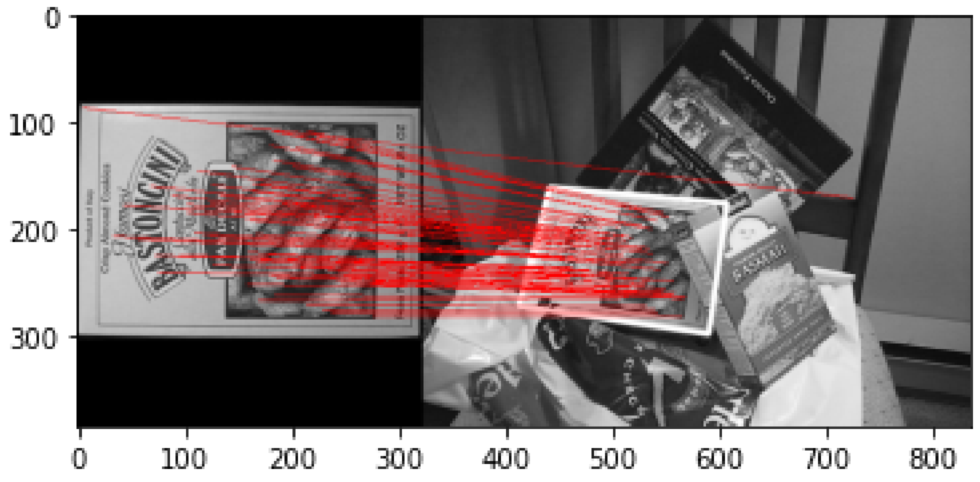
The source code is available here. The code contain following routines:

- generateBaseImage()
- computeNumberOfOctaves()
- generateGaussianKernels
- findScaleSpaceExtrema()
- generateDescriptors()

First, we call generateBaseImage() to appropriately blur and double the input image to produce the base image of our "image pyramid", a set of successively blurred and downsampled images that form our scale space. We then call computeNumberOfOctaves() to compute the number of layers ("octaves") in our image pyramid. Now we can actually build the image pyramid. We start with generateGaussianKernels() to create a list of scales (gaussian kernel sizes) that is passed to generateGaussianImages(), which repeatedly blurs and downsamples the base image. Next we subtract adjacent pairs of gaussian images to form a pyramid of difference-of-Gaussian ("DoG") images. We'll use this final DoG image pyramid to identify keypoints using findScaleSpaceExtrema(). We'll clean up these keypoints by removing duplicates and converting them to the input image size. Finally, we'll generate descriptors for each keypoint via generateDescriptors().

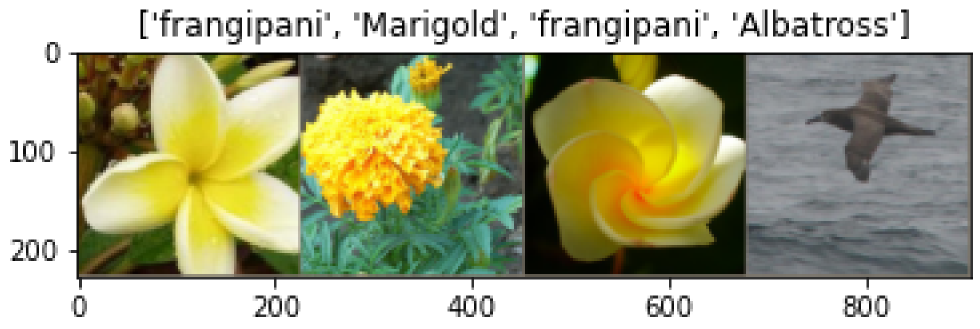
1.3 Detected keypoints

In the fig. given below, we can see the detected keypoints in two images.



2 Question 2

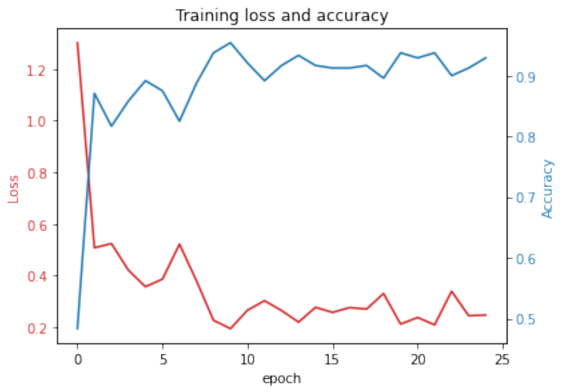
We are classifying the images for the give data with pre-trained ResNet model. The source code is available here. Sample images are as following:

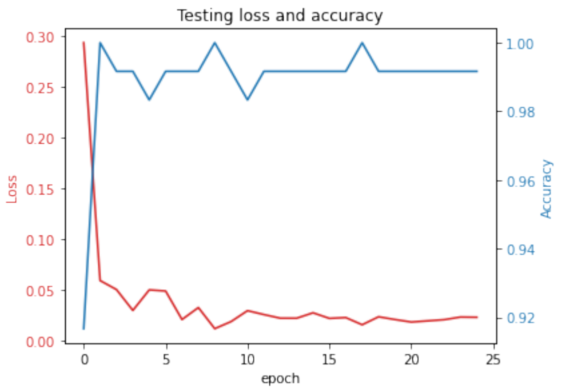


Part 1: We use features extracted from ResNet and use NN classifier.
Part 2: Fine tune the classification layer.

2.1 Part 1: Classification with features extracted from ResNet

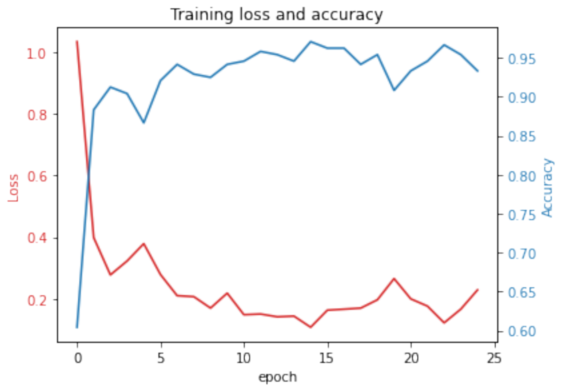
The training loss, accuracy and training loss, accuracy curves are shown below:

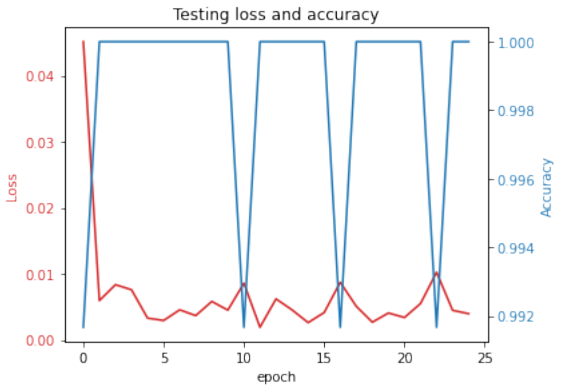




2.2 Part 2: Fine Tuning the ResNet

The training loss, accuracy and training loss, accuracy curves are shown below:





2.3 Comparison between part 1 and 2

- Training and testing losses are less in part 2.
- Training and testing accuracies are more in part 2.