Machine Learning

CV before DL

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Before Deep Learning

- I would like to give a glimpse about how communities where training models before deep learning
- The key effort in this stage was mainly on **extracting feature** vectors
- For example, assume we want to classify an image into: cat, dog, cow or horse
- How can we extract features for an image of 1024x1024 resolution?
- Let’s highlight Bag-of-features approach

Feature Descriptor

- Assume we have a small image batch of 40x40
- How can we represent this **visual content** as a feature vector in a robust and useful way?
  - So input is patch and output is e.g. feature vector of length 128
- 2 Major key aspects of feature descriptors:
  - **Robustness**: invariant to changes in illumination, noise, and geometric transformations like translation, rotation, and, to some extent, scale and perspective changes.
  - **Distinctiveness**: They should be able to match corresponding features in different images
  - Efficiency: They should be compact and fast to compute and compare, as they often need to be calculated in real-time applications

Well-known feature descriptors

- **SIFT** (Scale-Invariant Feature Transform): It transforms image data into scale-invariant coordinates relative to local features. It is robust to changes in rotation, scale, and illumination.
  - Another similar but faster one is SURF
- **HOG** (Histogram of Oriented Gradients): This descriptor counts occurrences of gradient orientation in localized portions of an image. It is particularly useful for human detection in images.
- **Others:** ORB and LBP

Feature Detector

- Now we know we can describe a patch
- But which patches to describe?
- We can divide the image to small mxm patches
  - However, many of them are not important for tasks
- There are algorithms called feature detectors that identify **points of interest** within an image (called features or keypoints), are **distinctive and easily recognizable areas**
  - For example areas in the image that have significant variation in brightness, color, or texture.
  - Detectors: Harris Corner Detector, SIFT, FAST, SURF, ORB, BRISK

Feature Detector

- Code wise, it is trivial. Typically we detect and describe important batches

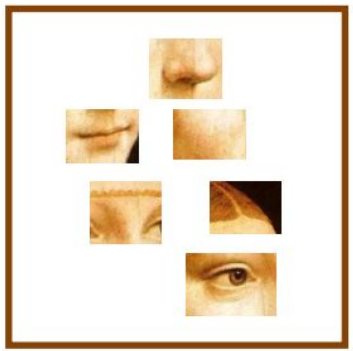
```
# Initialize SIFT detector
sift = cv2.SIFT_create()

# Detect SIFT features, also known as keypoints
keypoints, descriptors = sift.detectAndCompute(gray_image, None)

# Draw keypoints on the image
keypoint_image = cv2.drawKeypoints(image, keypoints,
                                   None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

Next?

- Now, we have e.g. 200k feature vectors representing the most important 200k locations in an image. Then what?!
- In theory, you can consider them another image representation
- But this is impractical
- We need a **fixed length** feature vector!
- We also observe, many of these vectors will be very close to each other (e.g. texture of similar colors?!

A visual word

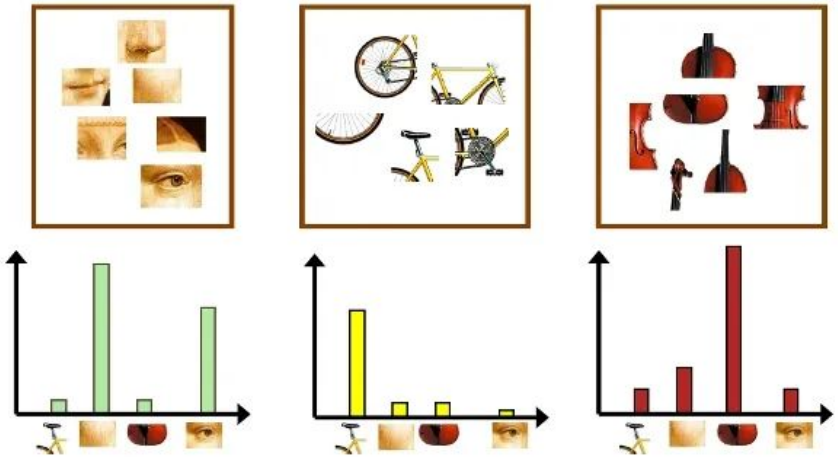
- Let’s cluster the 200k feature vectors into K clusters
- Now, each cluster centroid is a good representation for many patches
- Let’s call this centroid as a visual word
  - Think like NLP, a word consists of letters
  - A visual word is a representative visual patch / feature vector of visual content

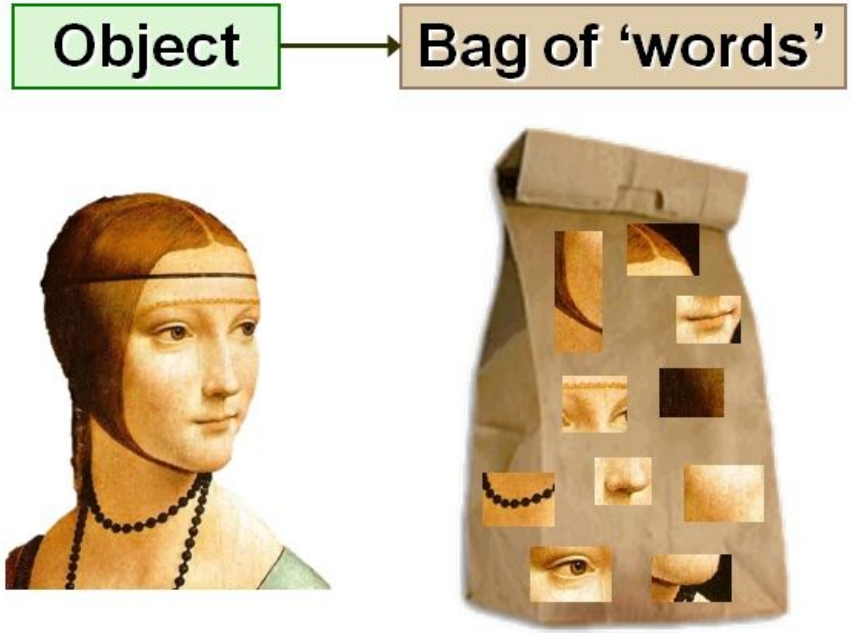


Bag of visual words

- Given a dictionary of K-words, and list of feature vectors of an image, we can for each feature vector, find its nearest cluster
- Now, build a histogram of these features counting the frequency of the clusters that they match
  - F1 ⇒ **C1**, F2 ⇒ C2, F3 ⇒ **C1**, F4 ⇒ **C1** ⇒ [C1 = 3, C2 = 1]: **Frequency**
- We call this histogram a bag of visual words
  - Idea inspired from NLP where we create a histogram of words frequency

Bag of visual words





Img src

Building a classifier

- Now, every image can be represented as a vector of length K
  - Representing the frequency of the visual words
- You have the data ready!
- Run a classical model on the data

Procedure

- Given N images
  - Detect and describe their features
  - Aggregate together
  - Cluster into K clusters [hyperparameter]
- Given a single image
  - Detect and describe features of the image
  - Find the cluster of each feature description
  - Create a histogram of the frequency
- Data is ready
  - Split and train

Procedure: Build histogram

- Refers to the full code if interested

```
# Function to build histograms of visual word occurrences
def build_histograms(images, kmeans):
    histograms = []
    for img in images:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        kp, des = sift.detectAndCompute(gray, None)
        if des is not None:
            pred = kmeans.predict(des)
            hist = np.histogram(pred, bins=np.arange(k + 1), density=True)[0]
            histograms.append(hist)
        else:
            histograms.append(np.zeros(k))
    return np.array(histograms)
```

Relevant Materials

- Bag of Visual Words(BoVW)
- Bag of Visual Words in a Nutshell
-

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."