MECS 6616

Computer Vision II: Keypoints, Features, Encodings

Spring 2020

Matei Ciocarlie

Regression and Classification on Images

1,000 x 1,000 Pixels

1M pixel / channel

3M pixel / image

$$\mathbf{x}_i \subset \Re^{3,000,000}$$

$$N = ?$$

Regressor / Classifier

Image class

Bounding box location

Per pixel class

Key Insight for Learning on Images

- The entire image contains a lot of redundant information
- Learning on images (before Deep Learning):
  - use human intuition to reduce the dimensionality of the input, without sacrificing information…
  - … then feed the result to classifier / regressor.
- Can we determine the relevant parts of an image, and feed just those to a classifier or regressor?
- There are likely hundreds of methods along these lines…
  - we will exemplify with just a few of of them.

If Regular Images Are Too Large...

- … perhaps we should just work with tiny images?
- Paper study: “*80 million tiny images: a large dataset for non-parametric object and scene recognition*”, by Antonio Torralba, Rob Fergus and William T. Freeman.

If Regular Images Are Too Large...

- How about we perform dimensionality reduction?
- PCA: Eigenfaces...

N = 3,023
d = 87 x 65 pixels = 5,655

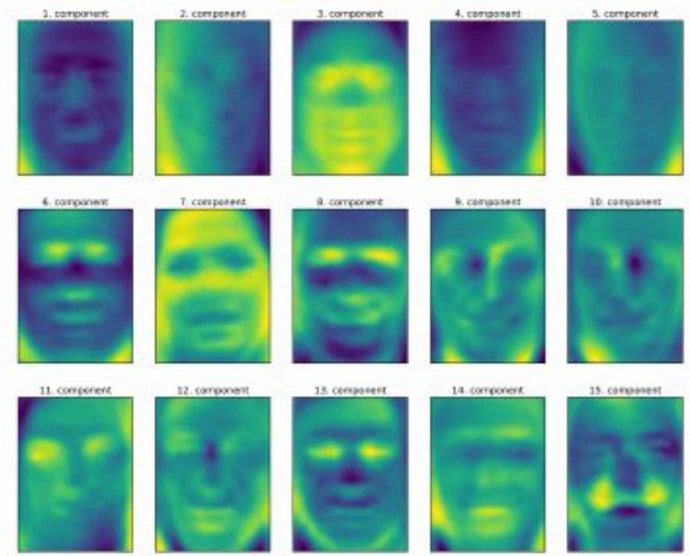
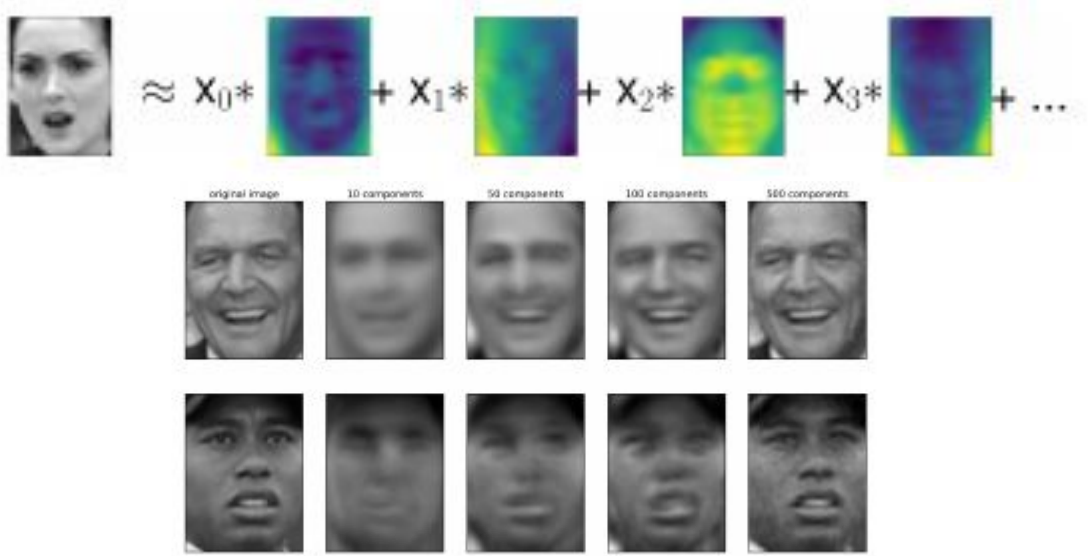


Figure 3-9. Component vectors of the first 15 principal components of the faces dataset



[Muller and Guido, Introduction to Machine Learning with Python]

If Regular Images Are Too Large...

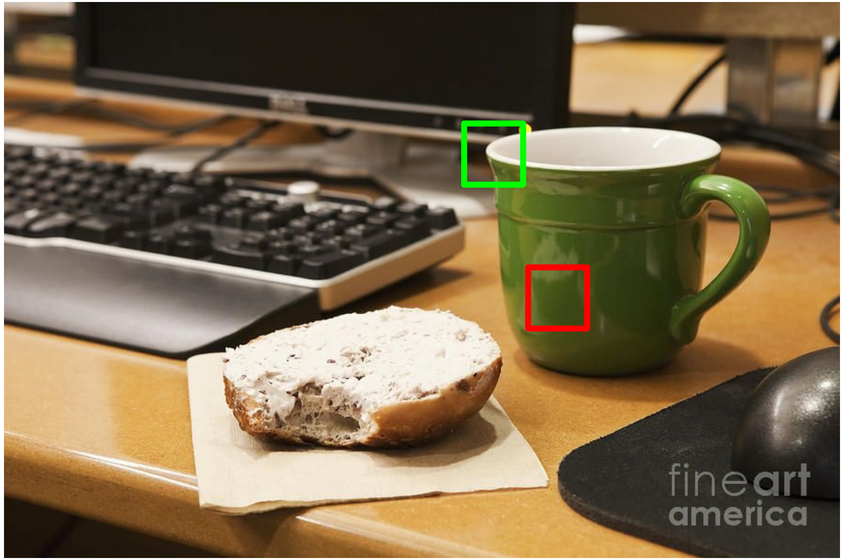
- How about we try to only use the best parts (patches) of the image?
- Example: Bag-of-Features classification
    - Goal: detect image class (e.g. “person”, “car”, “dog”, “mug”, etc.)

Bag-of-Features Image Classification

- Step 1: identify keypoints
  - Image points where "something interesting happens"
- One possible intuition:
  - Flat, uniform patches are bad…

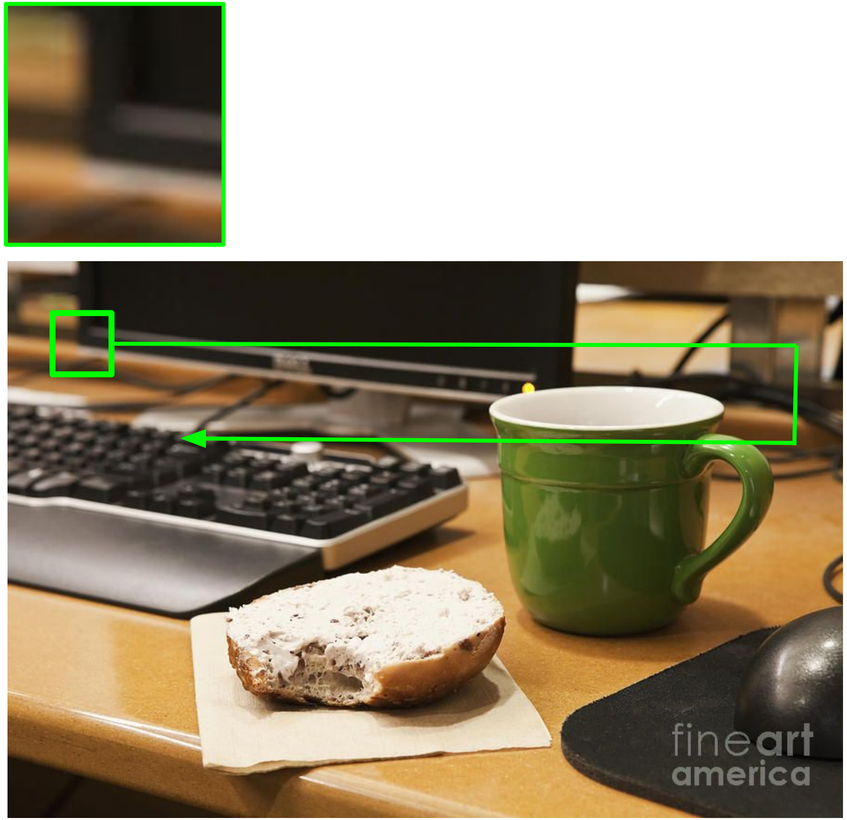
Bag-of-Features Image Classification

- Step 1: identify keypoints
  - Image points where "something interesting happens"
- One possible intuition:
  - Flat, uniform patches are bad…
  - … edges must be better…
  - … so corners are best!



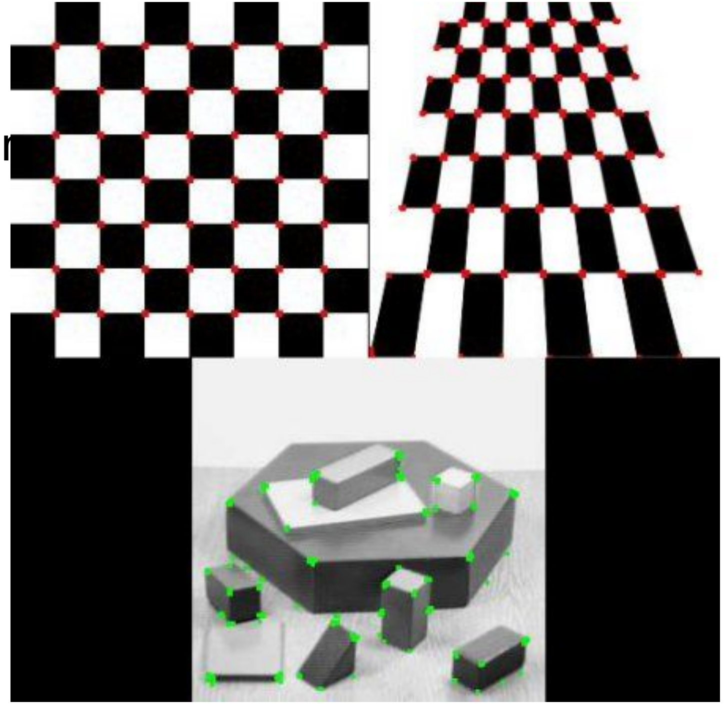
Bag-of-Features Image Classification

- Step 1: identify keypoints
  - Image points where “something interesting happens”
- One possible intuition:
  - Flat, uniform patches are bad…
  - … edges must be better…
  - … so corners are best!
- Detection:
  - “scan” the image
  - for each patch, decide if it contains a corner
  - must be fast!



Bag-of-Features Image Classification

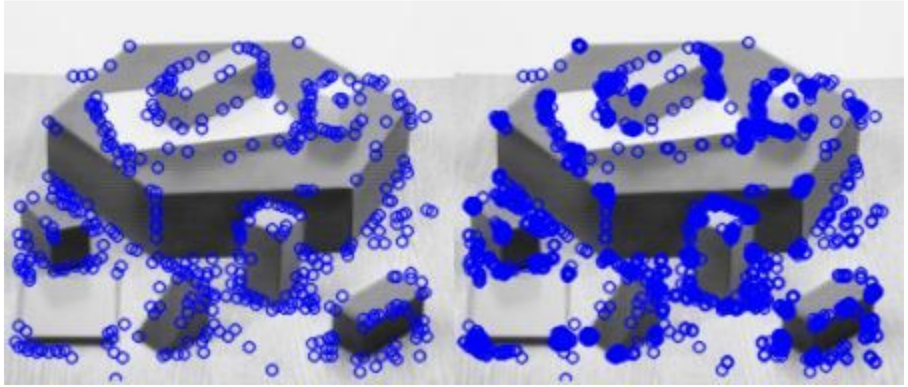
- Step 1: identify keypoints
  - Image points where “something interestin
- One possible intuition:
  - Flat, uniform patches are bad…
  - … edges must be better…
  - … so corners are best!
- Implementations:
  - Harris corner detector



[docs.opencv.org]

Bag-of-Features Image Classification

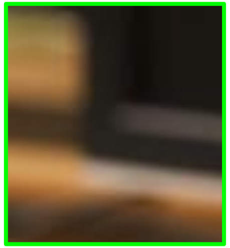
- Step 1: identify keypoints
  - Image points where “something interesting happens”
- One possible intuition:
  - Flat, uniform patches are bad…
  - … edges must be better…
  - … so corners are best!
- Implementations:
  - Harris corner detector
  - FAST keypoint detector
  - … many others



[docs.opencv.org]

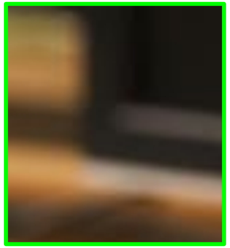
Bag-of-Features Image Classification

- Step 2: compute keypoint description
- Convert small (e.g. 16x16) image patch around keypoint into a feature vector



Bag-of-Features Image Classification

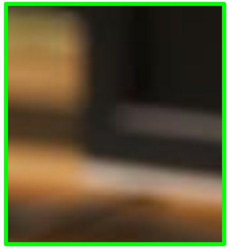
- Step 2: compute keypoint description
- Convert small (e.g. 16x16) image patch around keypoint into a feature vector



$$\begin{bmatrix} 12 \\ 23 \\ 61 \\ \dots \\ 99 \\ 10 \\ 02 \end{bmatrix}$$

Bag-of-Features Image Classification

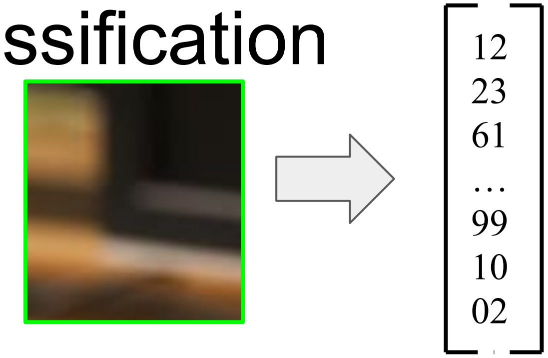
- Step 2: compute keypoint description
- Convert small (e.g. 16x16) image patch around keypoint into a feature vector
- Can I just stack the pixel values?

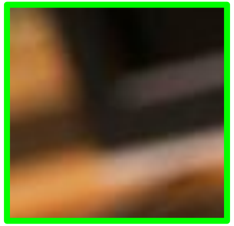


$$\begin{bmatrix} 12 \\ 23 \\ 61 \\ \dots \\ 99 \\ 10 \\ 02 \end{bmatrix}$$

Bag-of-Features Image Classification

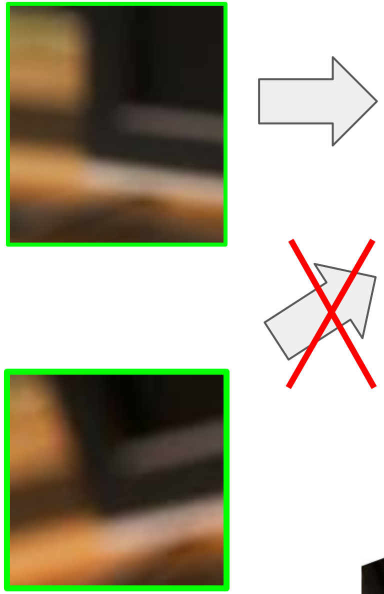
- Step 2: compute keypoint description
- Convert small (e.g. 16x16) image patch around keypoint into a feature vector
- Can I just stack the pixel values?





Bag-of-Features Image Classification

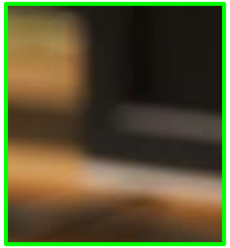
- Step 2: compute keypoint description
- Convert small (e.g. 16x16) image patch around keypoint into a feature vector
- Can I just stack the pixel values?
  - Bad idea: sensitive to changes in rotation, scale, focus, etc.



$$\begin{bmatrix} 12 \\ 23 \\ 61 \\ \ldots \\ 99 \\ 10 \\ 02 \end{bmatrix}$$

Bag-of-Features Image Classification

- Step 2: compute keypoint description
- Convert small (e.g. 16x16) image patch around keypoint into a feature vector

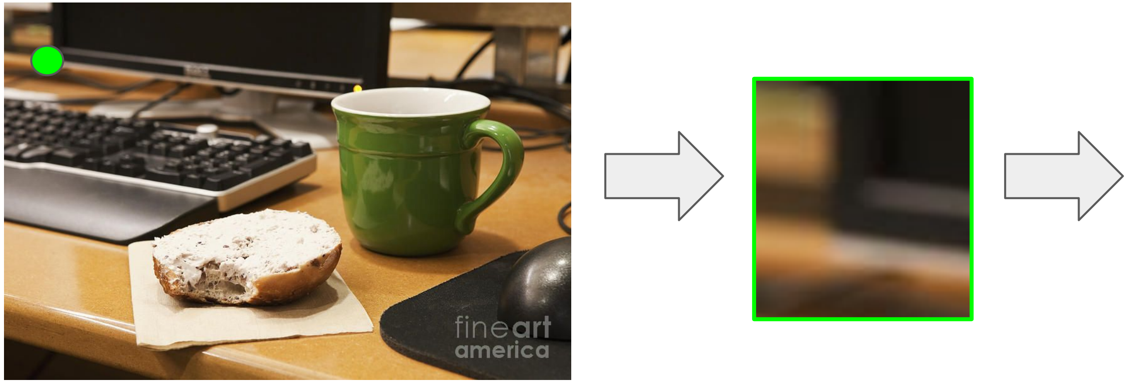


$$\begin{bmatrix} 12 \\ 23 \\ 61 \\ \ldots \\ 99 \\ 10 \\ 02 \end{bmatrix}$$

- There are dozens (or maybe hundreds) of papers on computing good keypoint descriptors (also called feature descriptors)
  - sensitive to the shape and color of underlying scene
  - insensitive to changes that we'd like to ignore (orientation, size, focus)
  - reasonably small (e.g. 100 dimensions, give or take)
  - reasonably fast to compute
- Very popular option: SIFT feature (D.Lowe, "*Distinctive Image Features from Scale-Invariant Keypoints*", 2004)

Bag-of-Features Image Classification

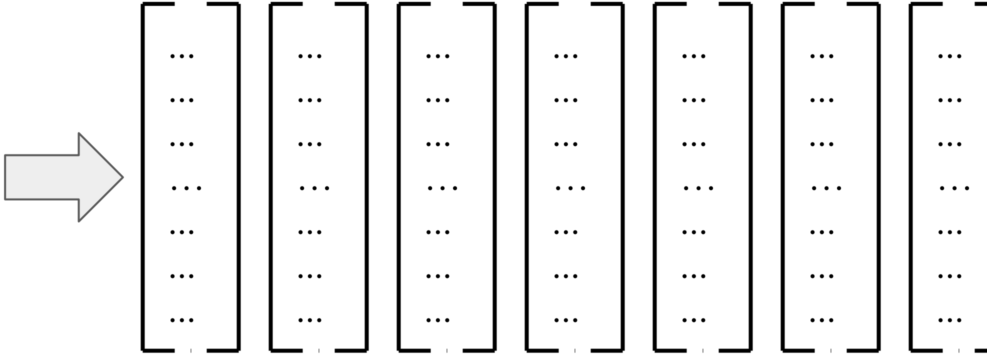
- Recap: we have ways to:
    - find an interesting patch in an image
    - encode it as a feature vector



$$\begin{bmatrix} 12 \\ 23 \\ 61 \\ \dots \\ 99 \\ 10 \\ 02 \end{bmatrix}$$

Bag-of-Features Image Classification

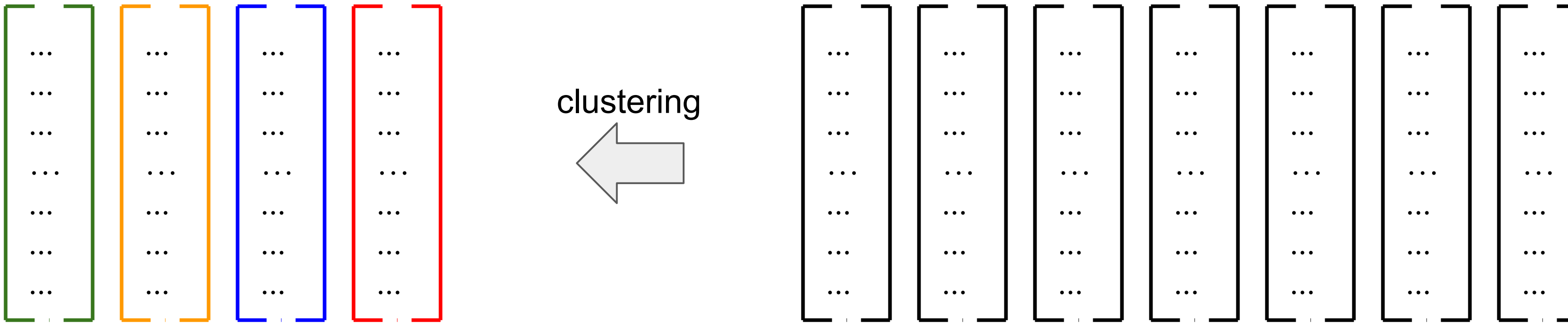
- Recap: we have ways to:
  - find an interesting patch in an image
  - encode it as a feature vector
- Step 3: do this for all images in our training set!



Lots and lots of feature vectors

Bag-of-Features Image Classification

- Recap: we have ways to:
  - find an interesting patch in an image
  - encode it as a feature vector
- Step 3: do this for all images in our training set!
- Step 4: cluster all feature vectors. Cluster reps form codeword dictionary.

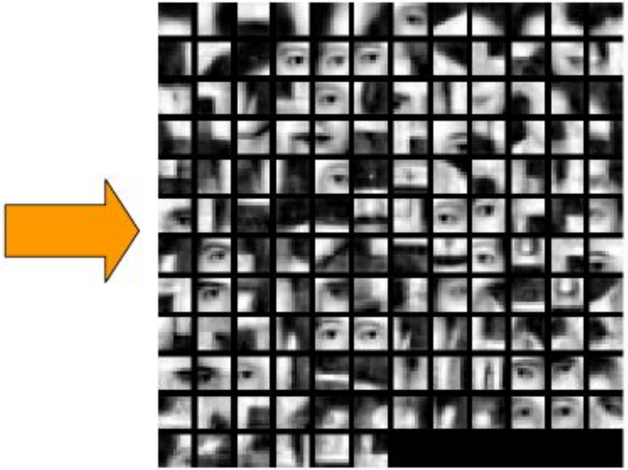


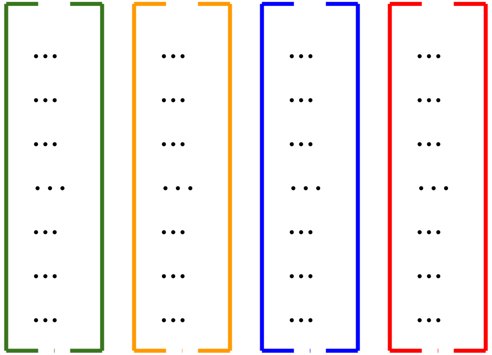
Codeword dictionary

Lots and lots of feature vectors

Bag-of-Features Image Classification

- Recap: the codeword dictionary tells us what features (descriptors of small image patches) are most commonly encountered in our training set



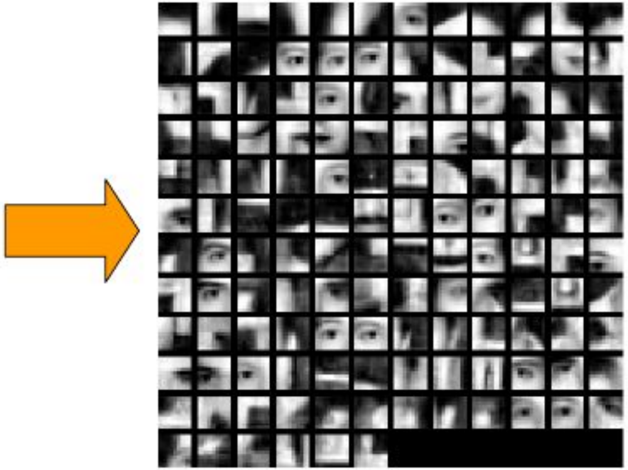


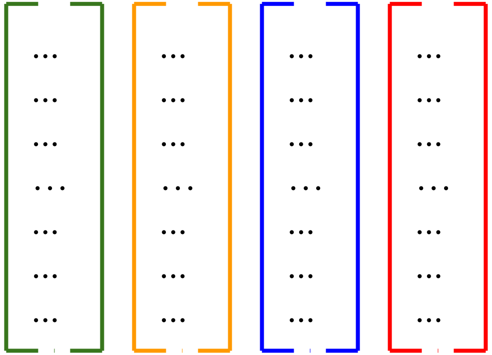
Codeword dictionary

[Andrew Zisserman, ICVSS 2008]

Bag-of-Features Image Classification

- Recap: the codeword dictionary tells us what features (descriptors of small image patches) are most commonly encountered in our training set
    - features are often referred to as visual words





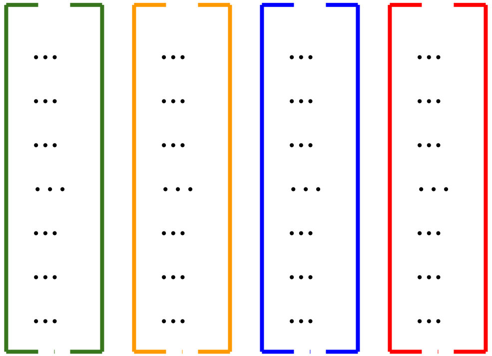
Codeword dictionary

[Andrew Zisserman, ICVSS 2008]

Bag-of-Features Image Classification

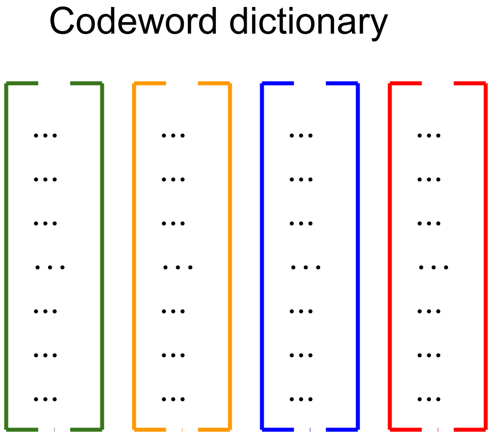
- Step 5: encode an image

Codeword dictionary



Bag-of-Features Image Classification

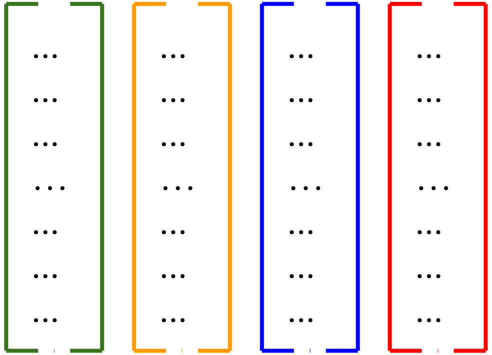
- Step 5: encode an image



Bag-of-Features Image Classification

- Step 5: encode an image

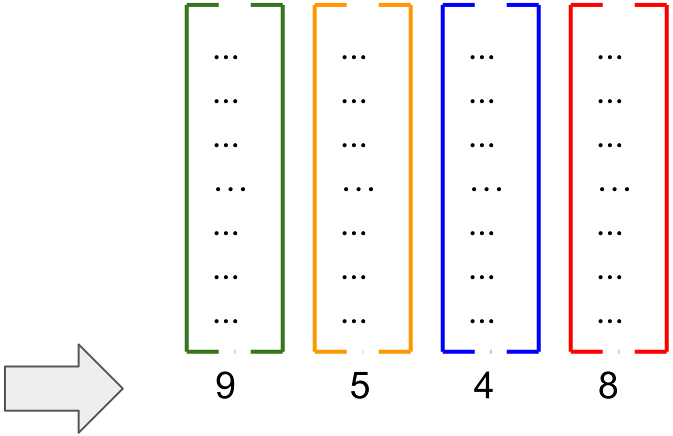
Codeword dictionary



Bag-of-Features Image Classification

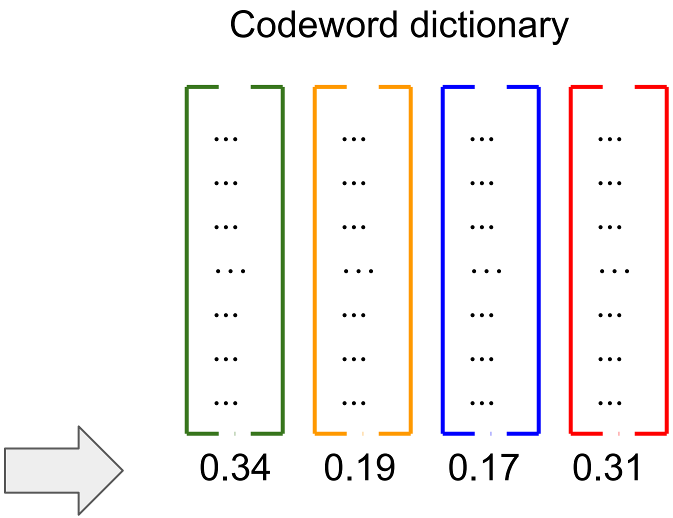
- Step 5: encode an image

Codeword dictionary



Bag-of-Features Image Classification

- Step 5: encode an image



Bag-of-Features Image Classification

- Step 5: encode an image

$$[\,0.34\,,\quad 0.19\,,\quad 0.17\,,\quad 0.31\,] \subset \Re^k$$

k = dictionary size

Bag-of-Features Image Classification

- Step 5: encode an image

This is my feature vector for this image!

$[\,0.34\,,\ \ 0.19\,,\ \ 0.17\,,\ \ 0.31\,] \subset \Re^k$

k = dictionary size

Bag-of-Features Image Classification

- Image is represented by how many of each feature it contains
- Any spatial relationship between features in the image is lost

$$[\,0.34\,,\ \ 0.19\,,\ \ 0.17\,,\ \ 0.31\,] \subset \Re^k$$

k = dictionary size

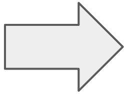
Bag-of-Features Image Classification

- Image is represented by how many of each feature it contains
- Any spatial relationship between features in the image is lost

=

Bag-of-Features Image Classification

- So far, we've just spoken about how to encode an image into a feature vector
- What about classification?



$\mathbf{x} \subset \Re^k$

k = dictionary size

Bag-of-Features Image Classification

- So far, we've just spoken about how to encode an image into a feature vector
- What about classification?

$\mathbf{x} \subset \Re^k, \mathbf{y} \subset \Re$

$\mathbf{x}_1 \rightarrow \text{"car"}$

$\mathbf{x}_2 \rightarrow \text{"cat"}$

...

$\mathbf{x}_n \rightarrow \text{"mug"}$

$\mathbf{x}_{new} \rightarrow ?$

- I can run my favorite classifier (e.g. kernel SVM) on the feature vectors...

Key Insight for Learning on Images

- Determine the relevant parts of an image
  - keypoint detector
- Use human intuition to reduce the dimensionality:
  - feature descriptor
  - cluster into codeword dictionary
  - take histogram of codeword frequency in image
- Feed the result to classifier / regressor
  - kernel SVM

Key Insight for Learning on Images

- Determine the relevant parts of an image
  - keypoint detector
- Use human intuition to reduce the dimensionality:
  - feature descriptor
  - cluster into codeword dictionary
  - take histogram of codeword frequency in image
- Feed the result to classifier / regressor
  - kernel SVM

What matters more, the encoding or the classifier?

Key Insight for Learning on Images

- Determine the relevant parts of an image
  - keypoint detector
- Use human intuition to reduce the dimensionality:
  - feature descriptor
  - cluster into codeword dictionary
  - take histogram of codeword frequency in image
- Feed the result to classifier / regressor
  - kernel SVM

What matters more, the encoding or the classifier?

Decades of debate… until Deep Learning.