Lecture Notes for

Machine Learning in Python

⟦ 🧑‍🏫, 🧑‍💻, 🐍, 🧑‍🔬 ⟧

Professor Eric Larson

Dimensionality Reduction and Images

Class Overview, by topic

Table Data Visualization
Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing
Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression
Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.
Numpy
Detailed mathematics for NN operations

Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**
Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models
ConceptNet
Case studies

Class Logistics and Agenda

- **Logistics**:
  - **Next Time**: Flipped Module
  - Do **quiz one** after this lecture!!
  - Turn in one per team (HTML), please include team member names from canvas
- **Agenda**
  - Common Feature Extraction Methods for Images
  - Begin Town Hall, if time

Class Overview, by topic

Table Data Visualization

Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing

Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression

Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.

Numpy
Detailed mathematics for NN operations

Wide and Deep Networks

Convolutional Networks

Recurrent Networks

Keras, Tensorflow
Intuition, Detailed implement.

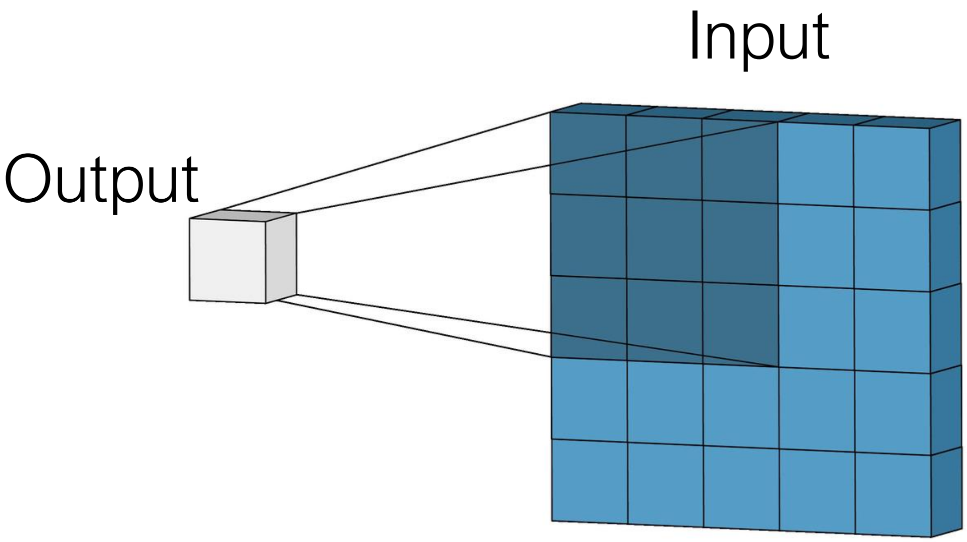
Ethics in Language Models

ConceptNet
Case studies

Features of Images

Extracting Features: Convolution

- For images:
  - kernel (matrix of values)
  - slide kernel across image, pixel by pixel
  - multiply and accumulate



This Example:
3x3 Kernel (dark)
Ignoring edges of input
Input Image is 5x5
Output is then 3x3

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution

$$\sum\left(\mathbf{I}\left[i \pm \frac{r}{2}, j \pm \frac{c}{2}\right] \odot \mathbf{k}\right) = \mathbf{O}[i,j]$$

output image at pixel i,j

input image slice centered in i,j with range r x c

kernel of size, r x c usually $r=c$

0	0	0	0	0	0	0	0	0
0	1	2	3	4	12	9	8	0
0	5	2	3	4	12	9	8	0
0	5	2	1	4	10	9	8	0
0	7	2	1	4	12	7	8	0
0	7	2	1	4	14	9	8	0
0	5	2	3	4	12	7	8	0
0	5	2	1	4	12	9	8	0
0	0	0	0	0	0	0	0	0

input image, $\mathbf{I}$

0	0	0
2	3	4
2	3	4

1	2	1
2	4	2
1	2	1

kernel filter, $\mathbf{k}$ 3x3

20	21	36	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…

output image, $\mathbf{O}$

Convolution Examples

Blur

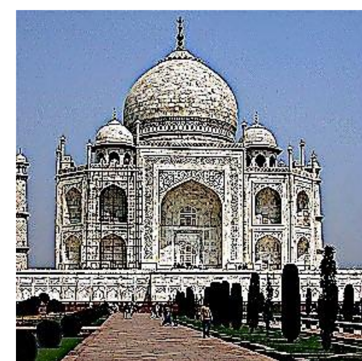
1	1	1
1	1	1
1	1	1

Vertical Edges

-1	0	1
-1	0	1
-1	0	1

Sharpen

0	-1	0
-1	5	-1
0	-1	0



0	0	0
1	0	0
0	0	0

Self test:
What does this do?

A. move left pixel to center
B. move right to center
C. blur

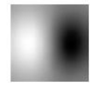
https://www.semanticscholar.org/paper/Automatic-Car-Damage-Recognition-using-Neural-Deijn/48f258246f7248a81458ddb14d9214e33759d7d7/figure/13

Common operations

- the gradient (2D derivative)

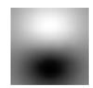
I

filter

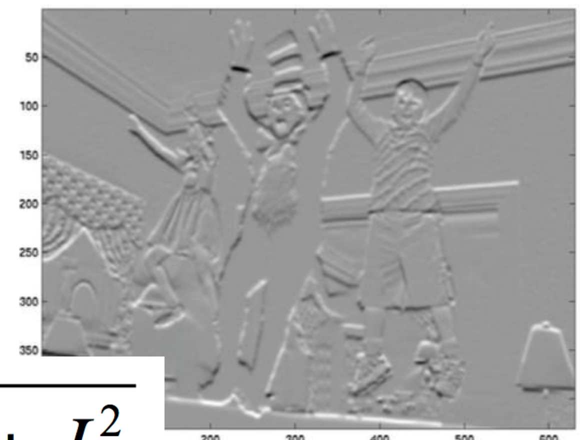


J_x

filter



J_y



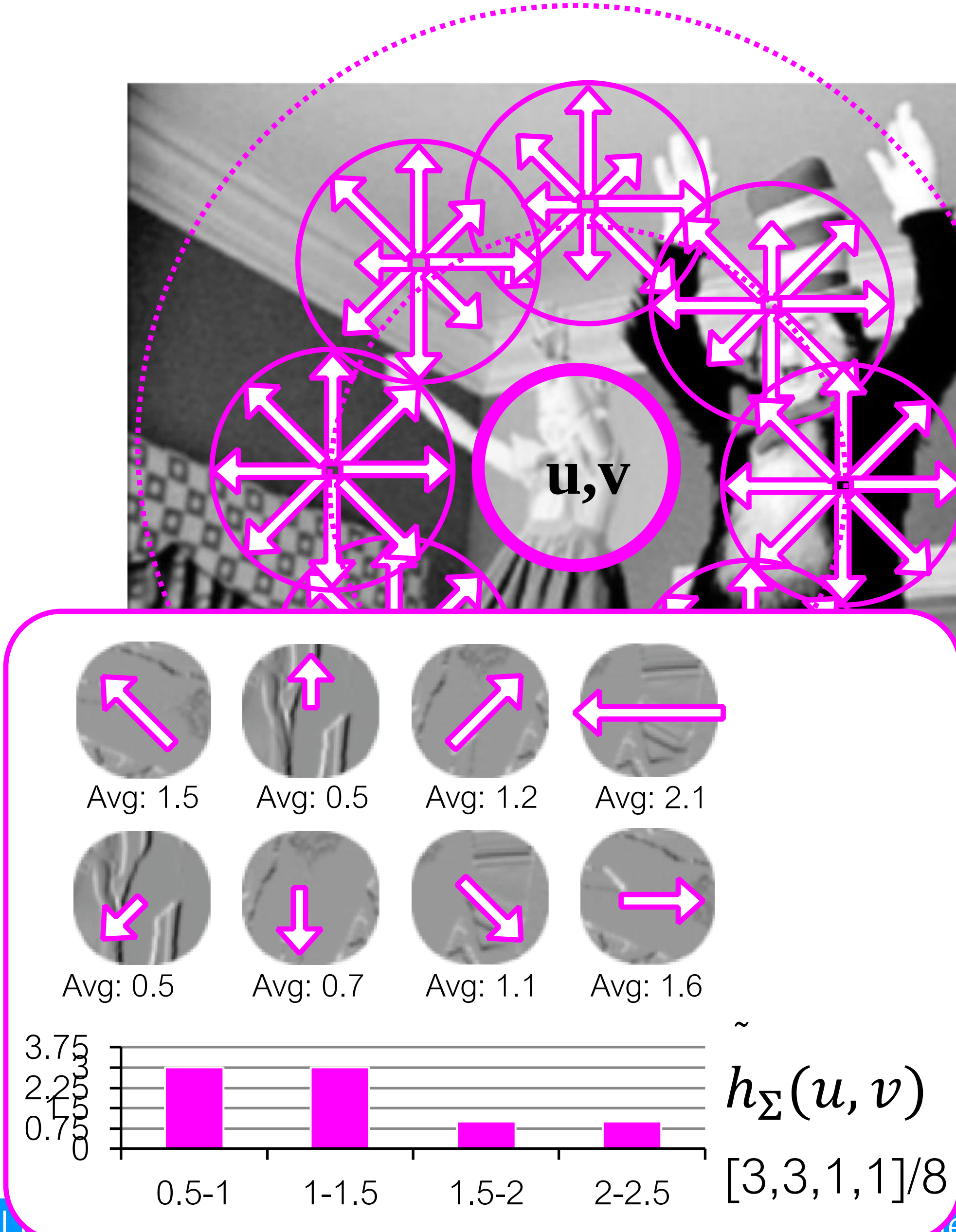
$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

$$O = \tan^{-1}(J_x / J_y)$$

J_y

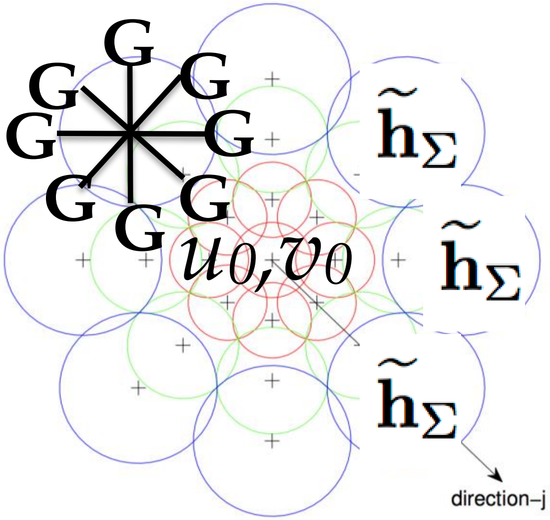
J_x

DAISY: same features, regardless of orientation



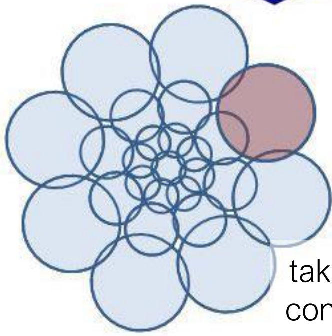
1. Select u,v pixel location in image and radius
2. Take histogram of average gradient magnitudes in circle for each orientation $\tilde{h}_\Sigma(u, v)$
3. Select circles in a ring, R
4. For each circle on the ring, take another histogram $\tilde{h}_\Sigma(\mathbf{l}_O(u, v, R_1))$
5. Repeat for more rings
6. Save all histograms as "descriptors" $[\tilde{h}_\Sigma(\cdot), \tilde{h}_\Sigma(\mathbf{l}_1(\cdot, R_1)), \tilde{h}_\Sigma(\mathbf{l}_2(\cdot, R_1)) \ldots]$
7. Can concatenate descriptors as "feature" vector at that pixel

Efficient DAISY, Orient x Circle Radius convolutions



one convolution per orientation

one convolve per ring size



take histogram of convolved images at points u,v

Daisy Operator at u_0,v_0 is Concatenated ||Histograms||

$$\mathcal{D}(u_0, v_0) =$$

$$\Big[\ \widetilde{\mathbf{h}}^\top_{\Sigma_1}(u_0, v_0),$$

$$\widetilde{\mathbf{h}}^\top_{\Sigma_1}(\mathbf{l}_1(u_0, v_0, R_1)), \cdots, \widetilde{\mathbf{h}}^\top_{\Sigma_1}(\mathbf{l}_T(u_0, v_0, R_1)),$$

$$\widetilde{\mathbf{h}}^\top_{\Sigma_2}(\mathbf{l}_1(u_0, v_0, R_2)), \cdots, \widetilde{\mathbf{h}}^\top_{\Sigma_2}(\mathbf{l}_T(u_0, v_0, R_2)),$$

take **normalized** histogram of magnitudes

$$\widetilde{\mathbf{h}}_\Sigma(u, v) = \left\| \left[\mathbf{G}_1^\Sigma(u, v), \ldots, \mathbf{G}_H^\Sigma(u, v)\right]^\top \right\|$$

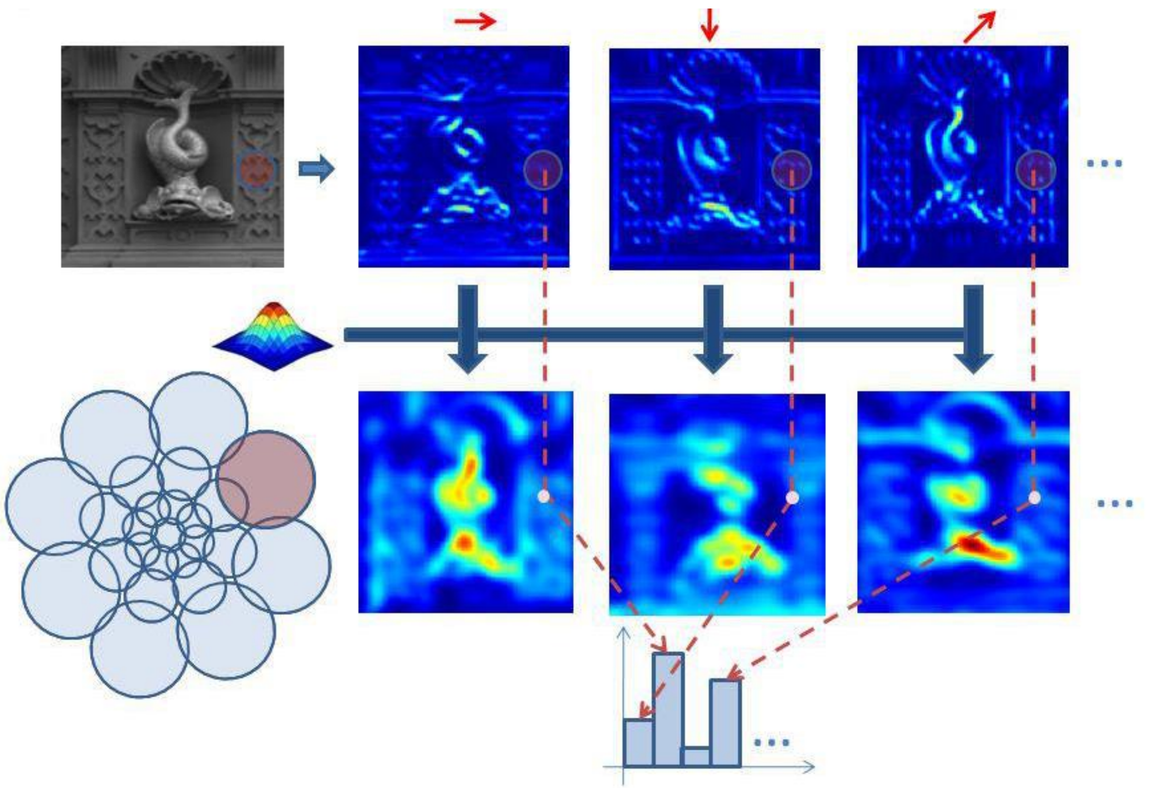
Tola et al. "*Daisy: An efficient dense descriptor applied to wide- baseline stereo.*" Pattern Analysis and Machine Intelligence, IEEE

An intuition for the future: DAISY workflow

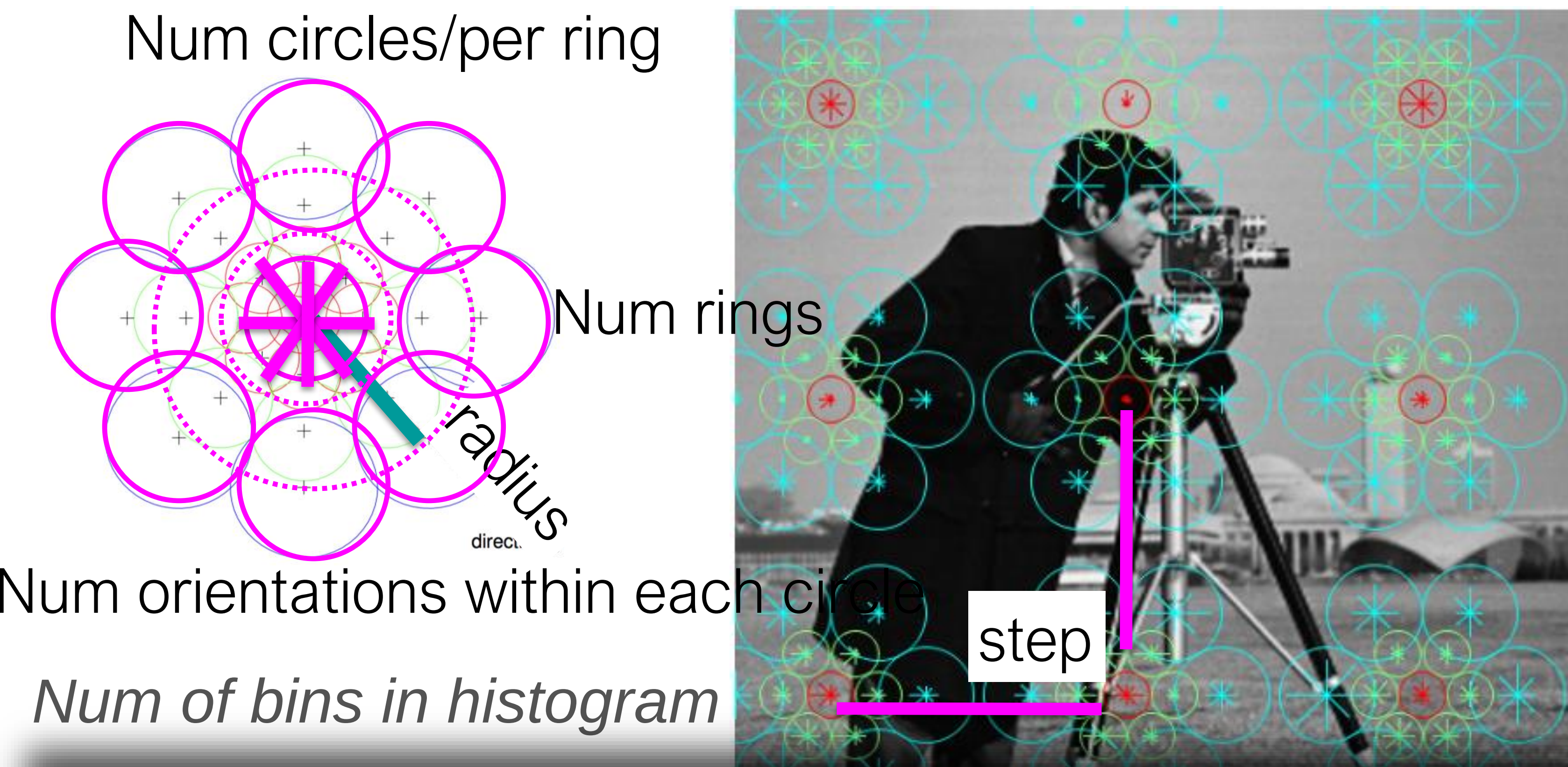
Convolutions on Image

Convolutions of Previous Convolutions

Statistics of Convolution Magnitudes



Hyper Parameters in DAISY, need selection




```python
daisy(img, step=180, radius=58, rings=2, histograms=6,
      orientations=8, visualize=True)
```

Params step, radius, num rings, num histograms per ring, orientations, *bins per histogram*

More Image Processing

Demo

Gradients

DAISY

(if time)Gabor Filter Banks

Other Tutorials:

http://scikit-image.org/docs/dev/auto_examples/

Matching versus Bag of Features

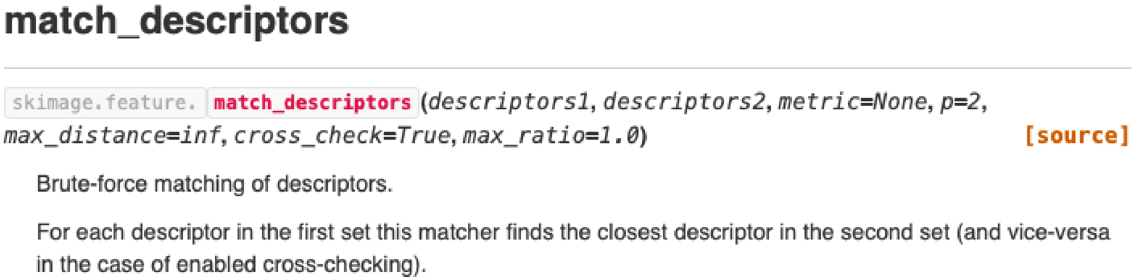
- Not a difference of vectors, but a percentage of matching points

- SURF, ORB, SIFT, DAISY

Matching test image to source dataset

1. Choose src image from dataset
2. Take keypoints of src image
3. Take keypoints of test image
4. For each kp in src:
    1. Match with closest kp in test
    2. *How to define match?*
5. Count number of matches between images
6. Determine if src and test are similar based on number of matches
7. Repeat for new src image in dataset
8. Once all images measured, choose best match as the target for the test image

Scikit-image Implementation



match_descriptors

`skimage.feature.match_descriptors(descriptors1, descriptors2, metric=None, p=2, max_distance=inf, cross_check=True, max_ratio=1.0)` [source]

Brute-force matching of descriptors.

For each descriptor in the first set this matcher finds the closest descriptor in the second set (and vice-versa in the case of enabled cross-checking).