BUILDING VISION SYSTEM USING MACHINE LEARNING

SCENE AND EVENT

Dr TIAN Jing

tianjing@nus.edu.sg

Module objective

Knowledge and understanding

- Understand the fundamentals of image feature learning for scene and event understanding

Key skills

- Design, build, implement and evaluate scene and event system for real-world application

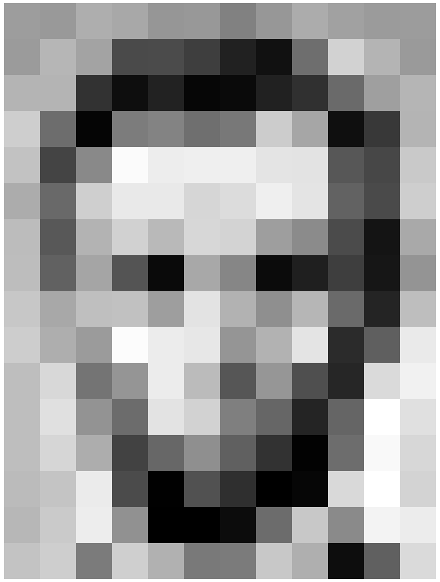
Major reference

- [Introduction] R. C. Gonzalez and R. E. Woods, ***Digital Image Processing***, http://www.imageprocessingplace.com/
- [Person re-identification] Theory and best practice, http://www.micc.unifi.it/reid-tutorial/
- [Person re-identification] FG 2018 tutorial, https://github.com/pkuvmc/pkuvmc.github.io/tree/master/FG2018-Tutorial/
- [Person re-identification] Deep Learning for Person Re-identification: A Survey and Outlook, https://arxiv.org/abs/2001.04193

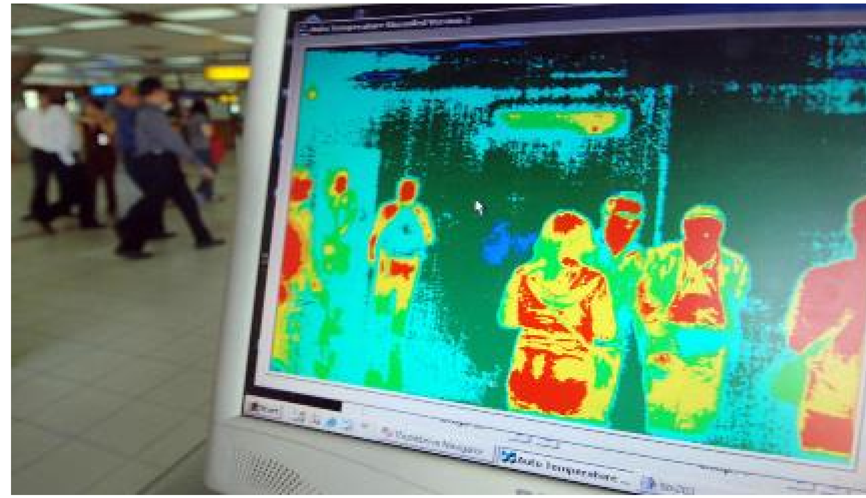
Topics

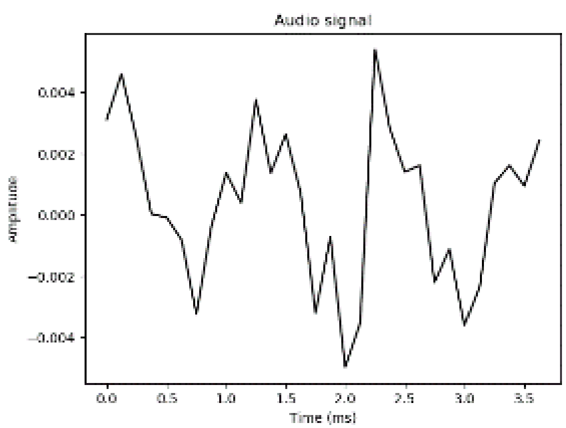
- Feature extraction and learning
- Person re-identification

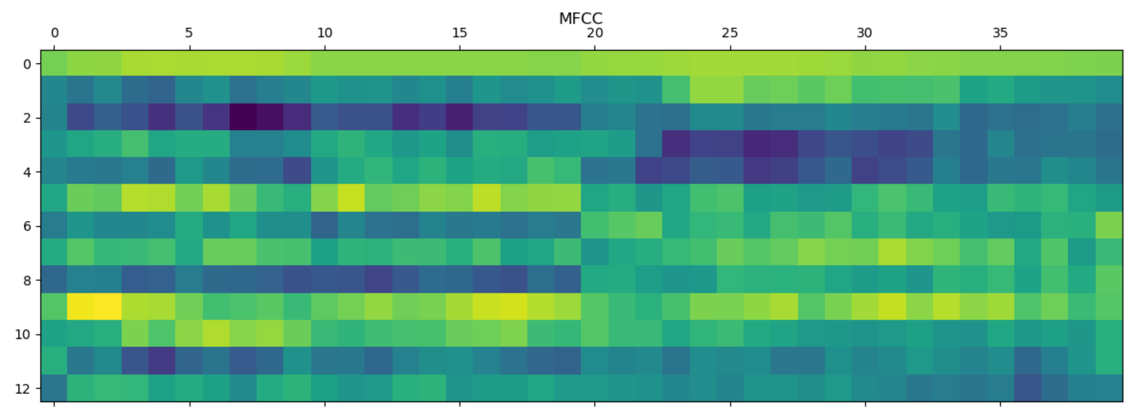
Image data



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218







Reference:

- http://techundred.com/how-snapchat-filter-work/
- https://www.tpr.org/post/thermal-imaging-gets-more-common-courts-havent-caught

Image understanding: Intuition (1)

Derived from the whole image

A: Summarize whole image into a single value

C: Keep information from each pixel

B: Use a set of values that are summarized over patches

Image understanding: Intuition (2)

Challenges of image understanding

Illumination | Scale | Rotation | Affine

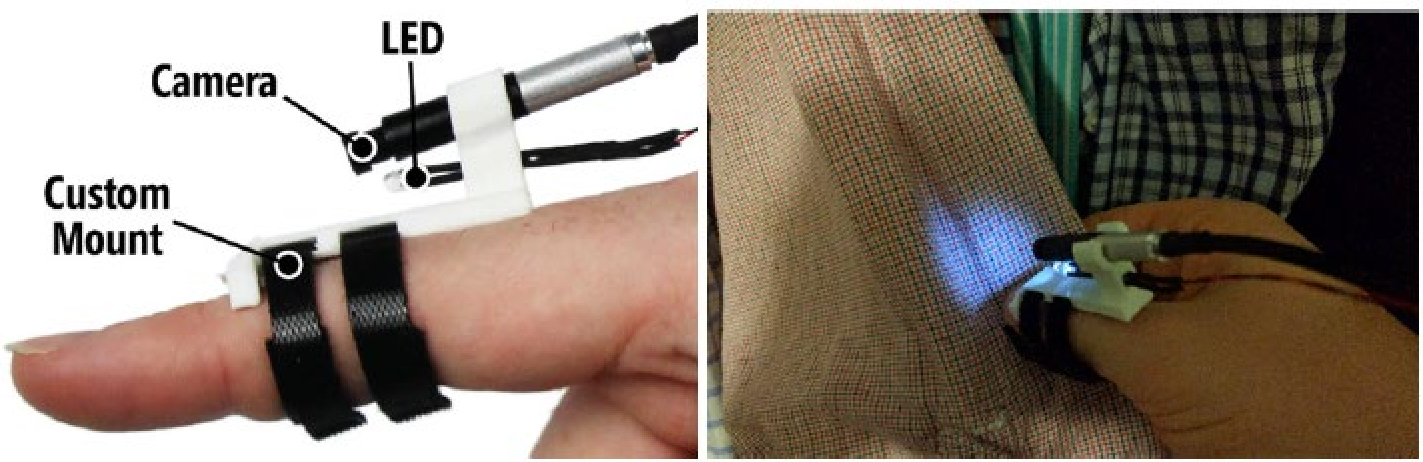
Overview of feature representation for image understanding

Category	Representative features
Color	Spectral peaks and histogram
Geometrical	Edges, lines, line widths, line relationships (e.g., parallel, perpendicular), circles, shapes, size of enclosed area
Statistical	Number of lines, area and perimeter, mean, variance, kurtosis, skewness, entropy
Time domain	Motion characteristics, speed, acceleration, trajectory
Frequency domain	Fourier coefficients and other time-frequency domains (such as discrete Cosine transformation, Gabor, Wavelet)

Texture

- Texture is characterized by the repetition of basic elements or *textons*.

Figure 1. Examples of the six classes in our fabric pattern dataset: solid, striped, checkered, dotted, zigzag, and floral.

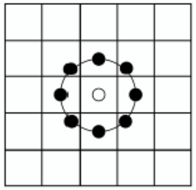


Reference: L. Stearns, L. Findlater, J. E. Froehlich, "Applying Transfer Learning to Recognize Clothing Patterns Using a Finger-Mounted Camera," Proceedings of ASSETS 2018, https://makeabilitylab.cs.washington.edu/project/clothrecognition/

LBP: Local binary pattern (1)

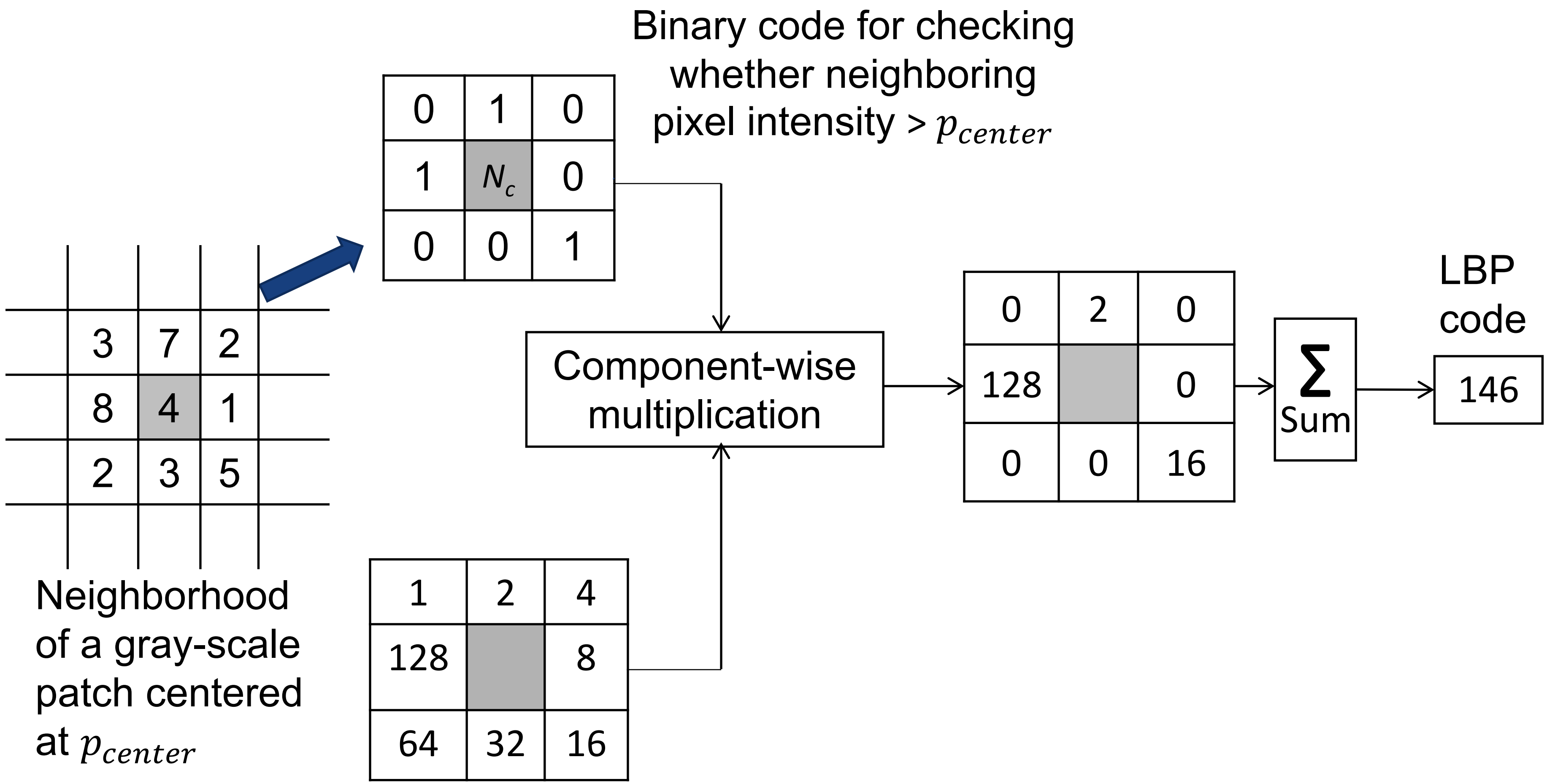
- For each pixel, compare the pixel to each of its (eight) neighbours (on its left-top, left-middle, left-bottom, right-top, etc.), use label “1” if the center pixel's value is smaller than the neighbour; otherwise, use label “0”. Given a set of elements $P = \{p_{center}, p_0, p_1, \cdots p_7\}$, where p_{center} represents the value of the central position, and $p_i (0 \leq i \leq 7)$ represent the values of its 3×3 neighbourhood. They can be characterized by a set of binary values $d_i (0 \leq i \leq 7)$ where

$$d_i = \begin{cases} 1 & \text{if } p_i \geq p_{center} \\ 0 & \text{if } p_i < p_{center} \end{cases}$$



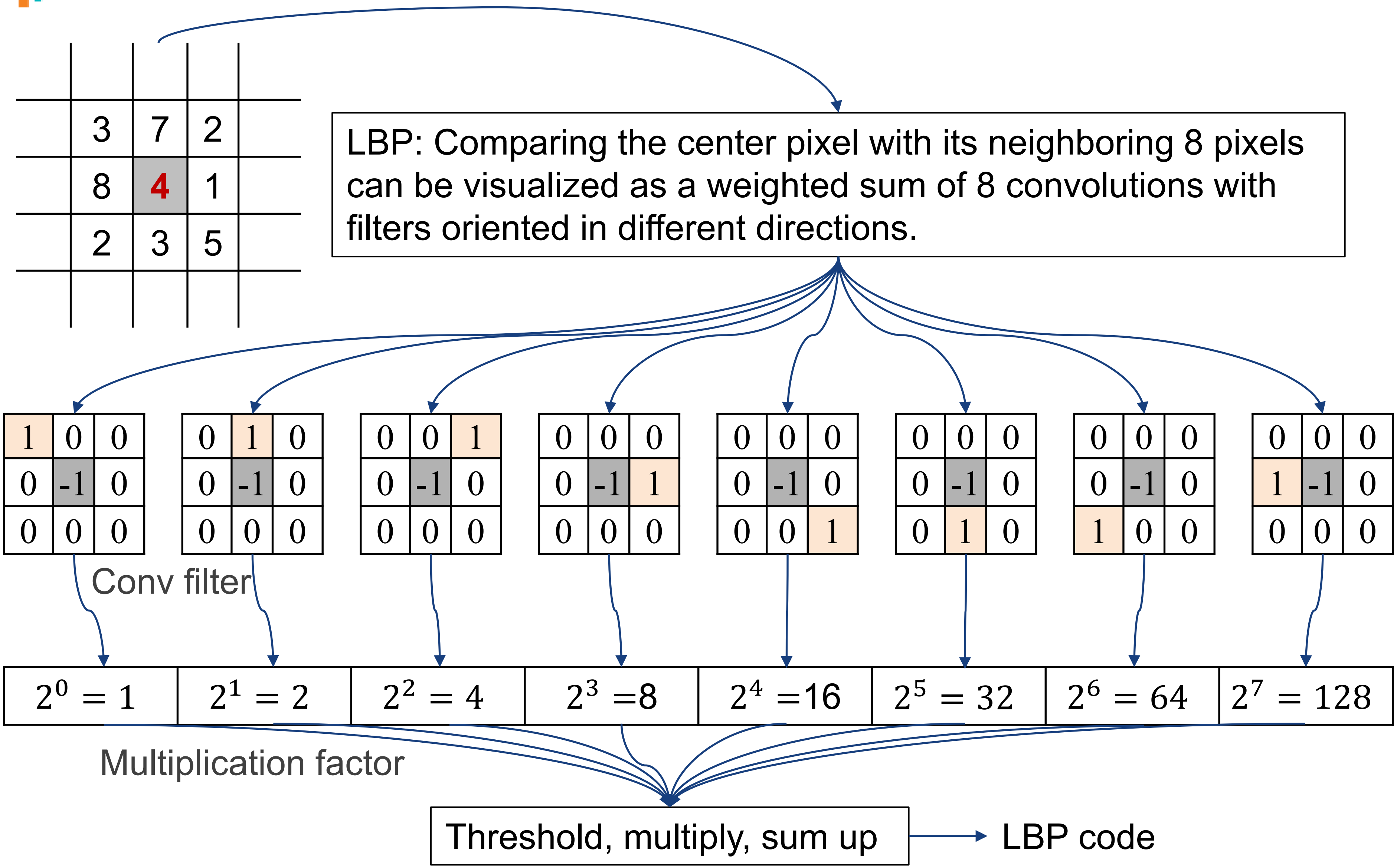
- The LBP code (the decimal value) for this pixel p_{center} is $LBP = \sum_{i=0}^{7} d_i \cdot 2^i$.
- Finally, compute the histogram of LBP codes for all pixels positions of the image.

LBP: Local binary pattern (2)



Template used to convert from binary number to decimal number. (Note: It is okay to use either clock-wise or counter clock-wise, as long as it is consistent).

LBP: Local binary pattern (3)

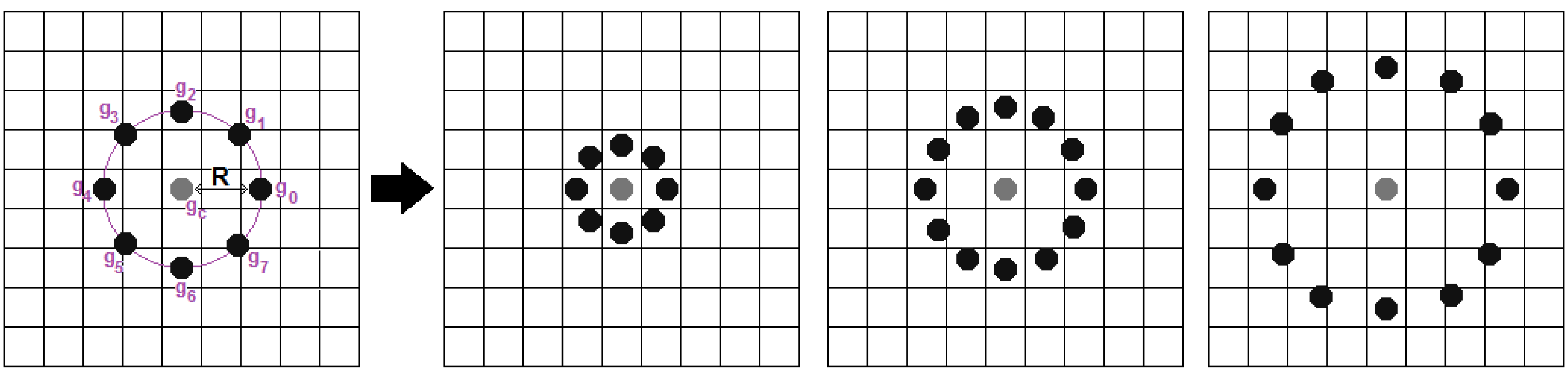


LBP: Local binary pattern (4)

A summary of the calculation of local binary pattern for an image.

1. Divide the image into cells (e.g., 16×16 pixels). Build one histogram per cell, then average all histograms to form a final histogram for the entire image.
2. For each pixel in a cell, compare it with its neighbours (various configurations are shown below). Where the neighbour pixel's value is greater than that of the center, score it as 1; otherwise score it as 0. Walking round all the neighbours then gives an array of binary numbers, and further convert it into decimal number.
3. For each cell, compute the histogram of the frequency of each decimal number. Optionally normalise the histogram.
4. Finally, average all the histograms to give a descriptor for the entire image.

Various configurations (P, R). P: Number of neighbours, R: radius between neighbours and the center pixel.



P=8, R=2 | P=8, R=1 | P=12, R=2 | P=12, R=3

HoG: Histogram of oriented gradients (1)

Clarification between Edge and Gradient		
Edge detection	Binary (yes/no)	Decision of edge detection task
Gradient	Continuous measurement	Used as features for other tasks

Gradient magnitude

$$m(x,y) = \sqrt{\underbrace{(I(x+1,y) - I(x-1,y))^2}_{\text{gradient in } x \text{ direction using filter } [-1,0,1]} + \underbrace{(I(x,y+1) - I(x,y-1))^2}_{\text{gradient in } y \text{ direction using filter } [-1,0,1]^T}}$$

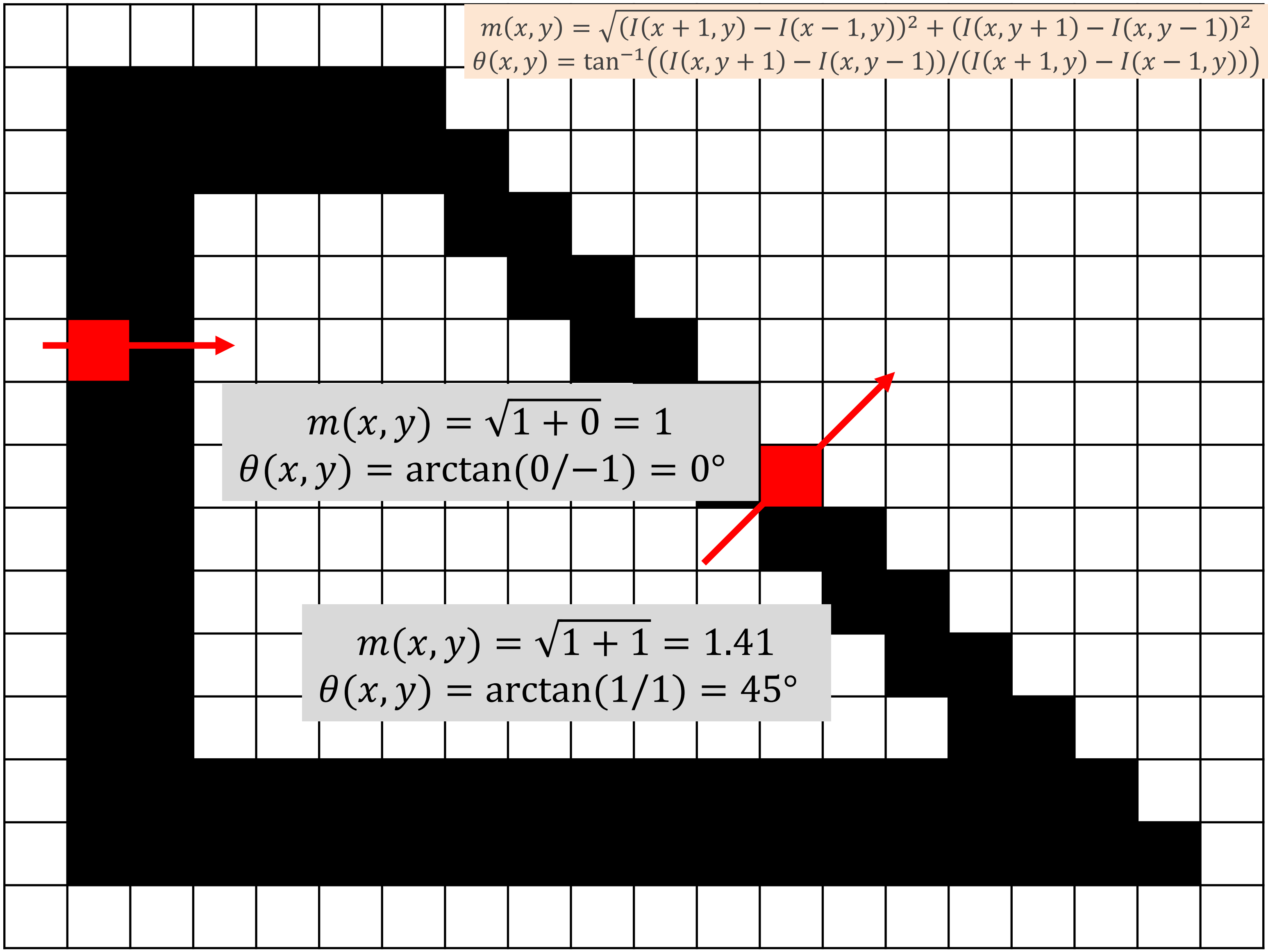
Gradient direction

$$\theta(x,y) = \tan^{-1}\left(\underbrace{(I(x,y+1) - I(x,y-1))}_{\text{gradient in } y \text{ direction using filter } [-1,0,1]^T} / \underbrace{(I(x+1,y) - I(x-1,y))}_{\text{gradient in } x \text{ direction using filter } [-1,0,1]}\right)$$

$I(x,y)$ = the image intensity at the pixel position (x,y)
An example of the calculation is provided in the following slide.

HoG: Histogram of oriented gradients (2)

$$m(x,y) = \sqrt{(I(x+1,y) - I(x-1,y))^2 + (I(x,y+1) - I(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}\big((I(x,y+1) - I(x,y-1))/(I(x+1,y) - I(x-1,y))\big)$$

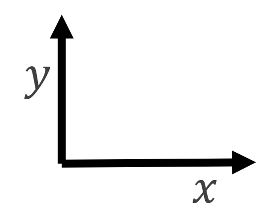


$$m(x,y) = \sqrt{1+0} = 1$$
$$\theta(x,y) = \arctan(0/-1) = 0°$$

$$m(x,y) = \sqrt{1+1} = 1.41$$
$$\theta(x,y) = \arctan(1/1) = 45°$$

Image gradient calculation

Black color: 0
White color: 1



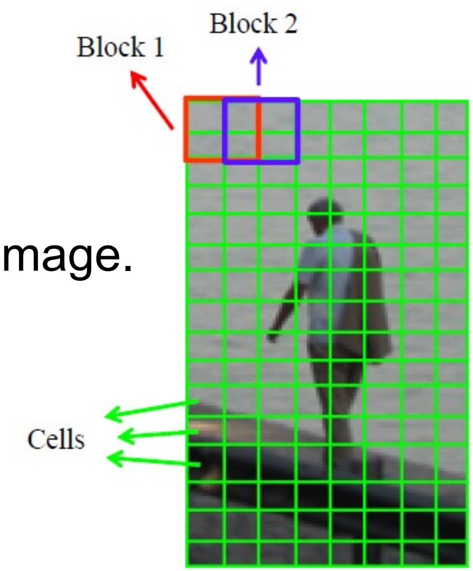
HoG: Histogram of oriented gradients (3)

Summary of HoG (suggested in the original CVPR2005 paper)

	Dimension	Remark
Image	64×128	A fixed-size input image
Block	16×16	With 50% overlap, Total 7×15=105 blocks
Cell	8x8	Each block should consist of 2×2 cells
Feature	9-bin	Each cell has a HoG feature of 9-bin histogram
Total features	3780	105 (block) × 4 (cell) × 9 (bin) = 3780

Given the fix-sized input image, build HoG for each *Cell*.
Note: How to handle input image with larger resolution?

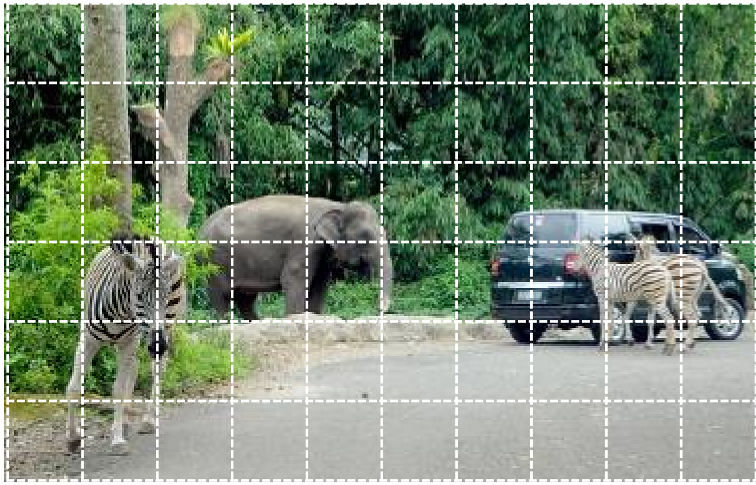
- The input image could be resized to be 64×128.
- The fixed-size window 64×128 could be sliding on the image.

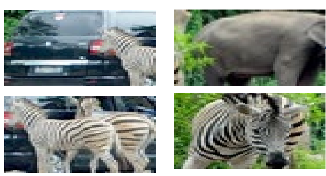


Reference: N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005, https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf.
https://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html

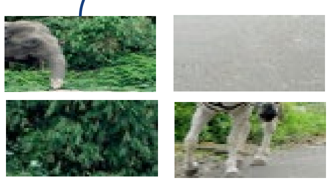
HoG: Histogram of oriented gradients (4)

Select sliding window (size, aspect ratio, stride), crop & resize





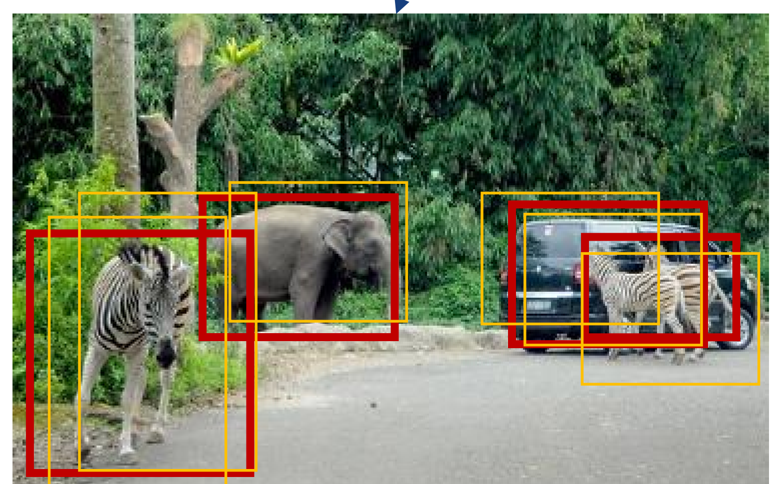
(offline) Train a HoG-based object classifier



Apply the classifier on each sliding window

Output: A set of
- Box label and score
- Box coordinates (x, y, w, h)

Apply *Non-Maximum Suppression* (NMS) to select best (red) boxes



Convolutional neural network (CNN)

- A CNN summary: https://cs231n.github.io/convolutional-networks/
- Excel tool: https://medium.com/apache-mxnet/multi-channel-convolutions-explained-with-ms-excel-9bbf8eb77108
- Visualization: https://www.cs.ryerson.ca/~aharley/vis/conv/flat.html
- Calculation of number of parameters: https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-by-hand-8f1716241889

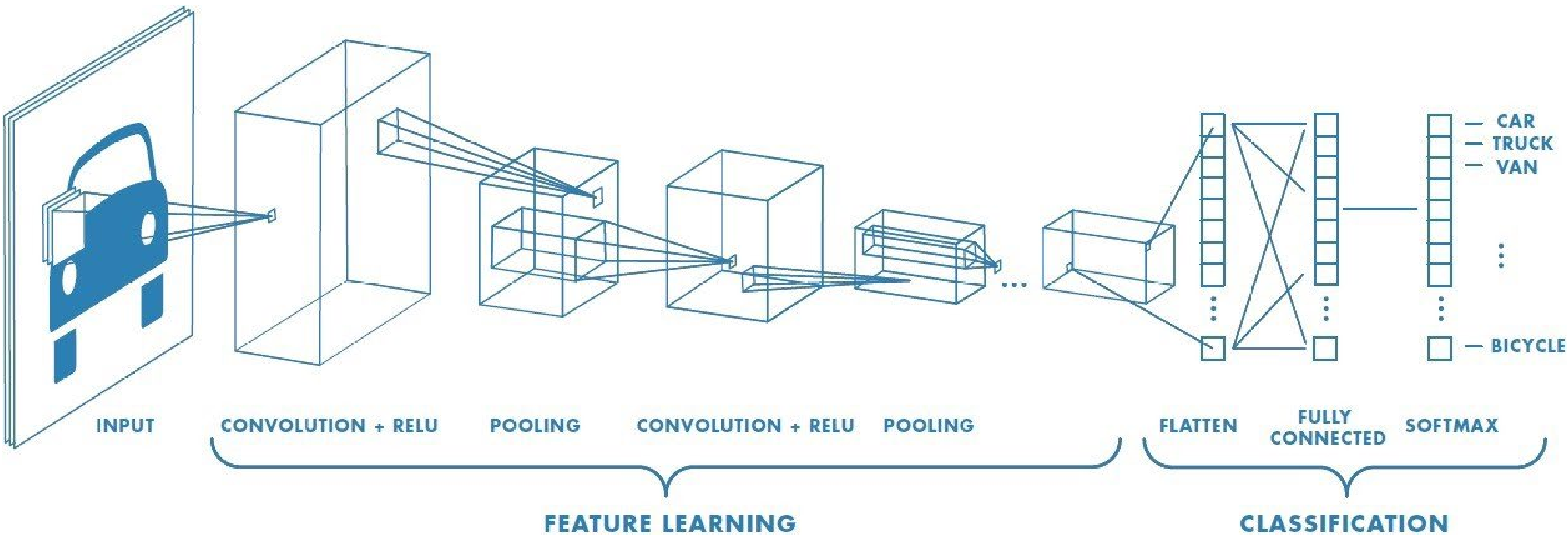
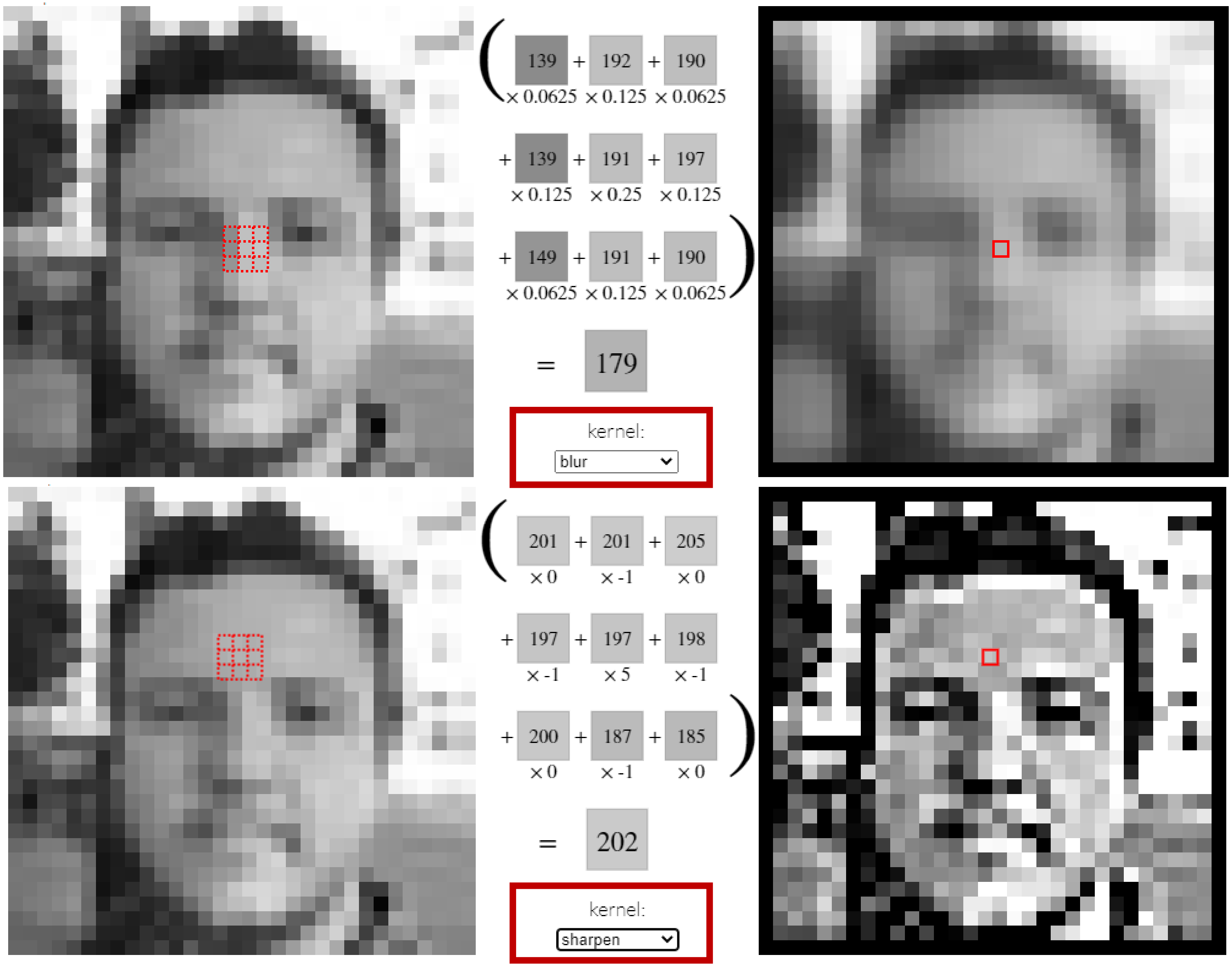


Photo: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CNN motivation: Image filter



Blur kernel

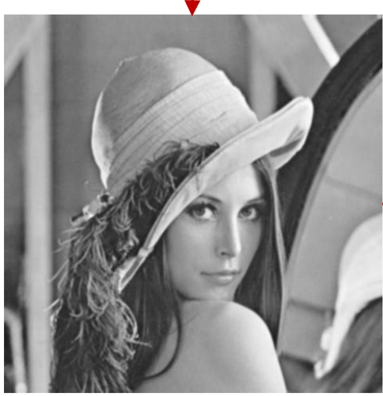
Sharpen kernel

Demo website: http://setosa.io/ev/image-kernels/

CNN motivation: Stacked filters

Input image

Apply Gaussian blur



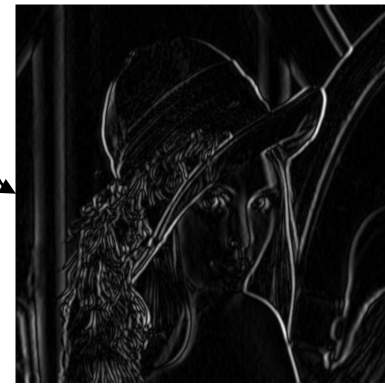
Convolution f_x

$$f_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

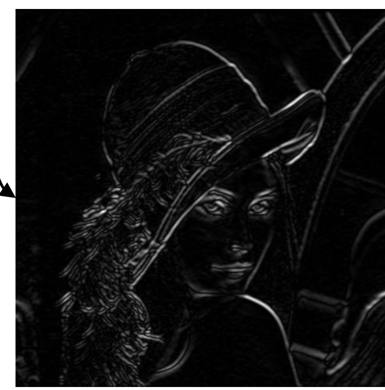
Convolution f_y

$$f_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Edge map G_x

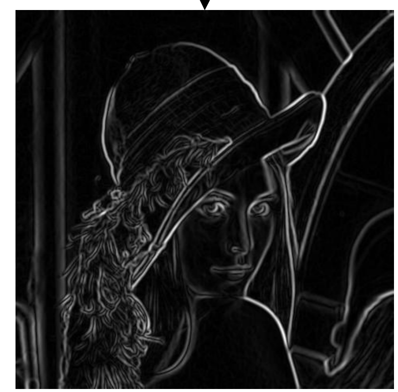


Edge map G_y



Calculate magnitude

$$G = \sqrt{G_x^2 + G_y^2}$$

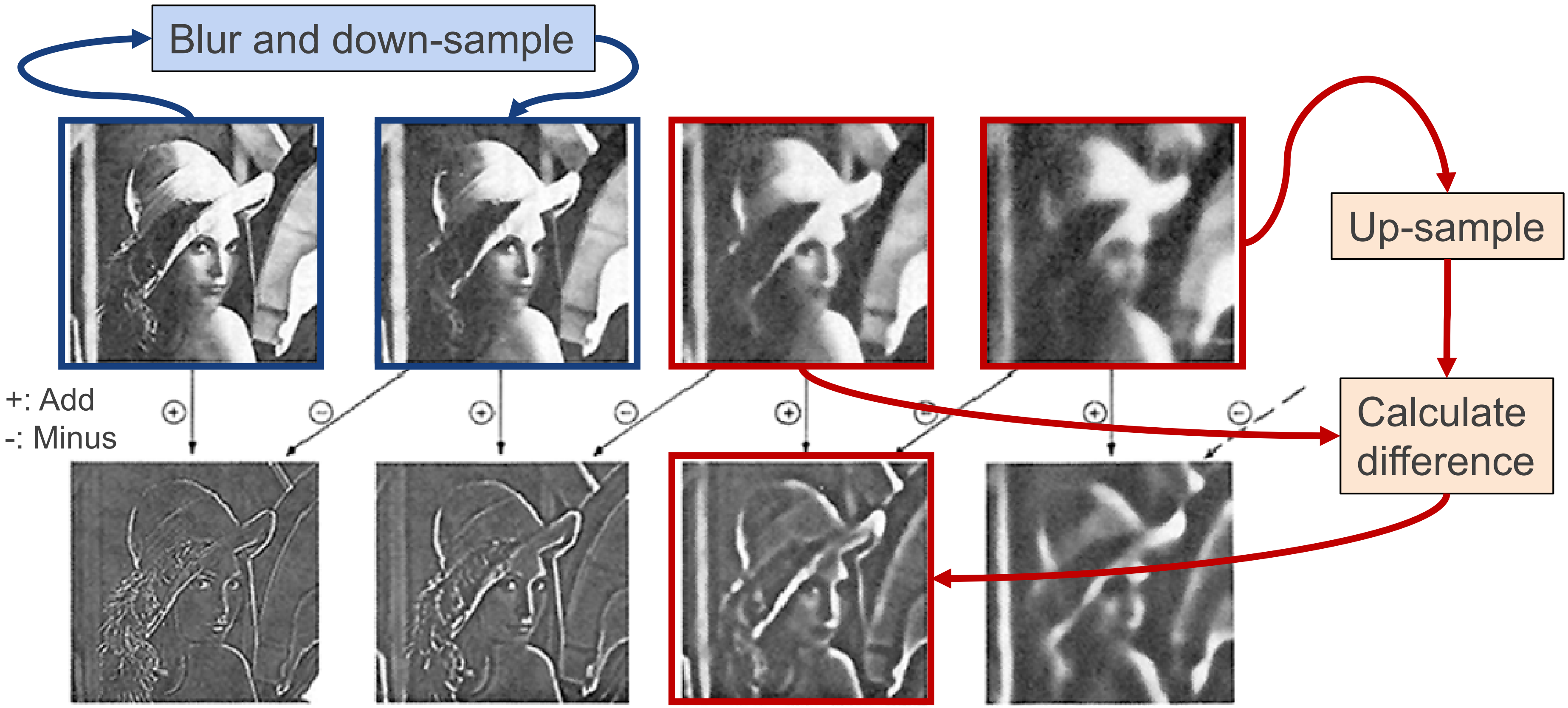


Final edge map

- Blue path: Sobel edge detection (non-stacked filter)
- Red path: Canny edge detection (stacked filter)

CNN motivation: Multiple-resolution stacked filters

- Gaussian pyramid: Progressively blurred and subsampled versions of the image.



- Laplacian pyramid: Compute the difference between up-sampled Gaussian pyramid level and Gaussian pyramid level.

Reference: The Laplacian Pyramid as a Compact Image Code, IEEE Trans. on Communications, 1983,http://persci.mit.edu/pub_pdfs/pyramid83.pdf

CNN as feature extractor

- ConvNet as a fixed feature extractor. Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. Since modern ConvNets take long time to train across multiple GPUs on ImageNet, it is common to see people release their final ConvNet checkpoints for the benefit of others who can use the networks for fine-tuning.
- Fine-tuning the ConvNet for your own dataset. Not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation. The earlier features of a ConvNet contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset.

Reference: https://cs231n.github.io/transfer-learning/

Topics

- Feature extraction and learning
- Person re-identification

Motivation: Person verification

Problem statement

- Recognition: Given a photo (face/body), classify among possible persons
- Verification: Verify that two photos belong to the same person

"Use **CCTV footage** to assist the government to do contact tracing" (Source: CNA interview https://www.youtube.com/watch?v=XrGtmcpjVrY)

Photo: https://github.com/pkuvmc/pkuvmc.github.io/tree/master/FG2018-Tutorial/

SINGAPORE

Contact tracing process under way: Health Minister

Ministry working to contact those who were in close proximity to Chinese national who had tested positive for Wuhan coronavirus

Source: https://www.todayonline.com/singapore/wuhan-virus-singapore-confirms-first-imported-case-another-suspected-case-has-positive, (24 January 2020)

Motivation: Person verification

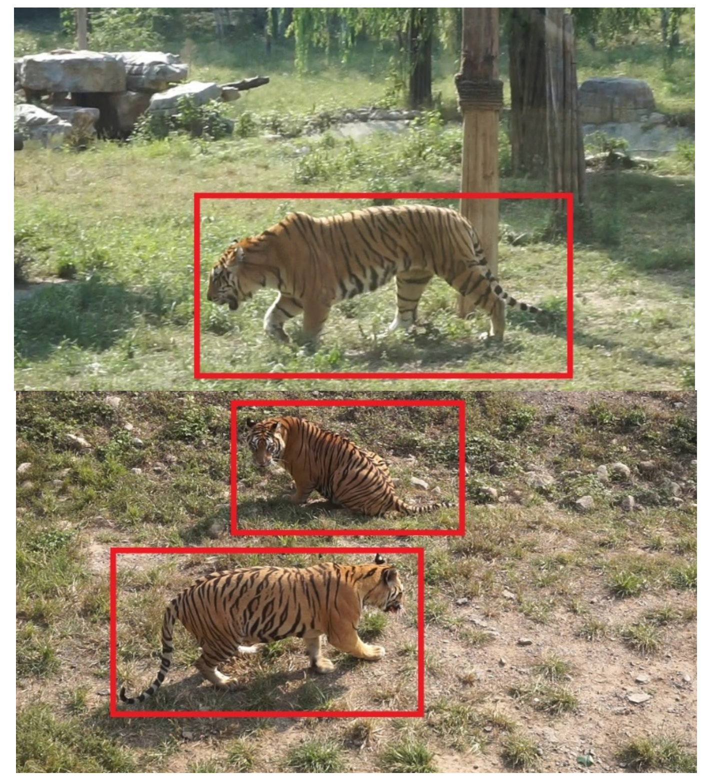
- Face: Frontal face? Sufficient resolution?
- Gait: Controlled or uncontrolled environment?
- Appearance: clothing color and texture? Hair style?

Photo:
- https://fortune.com/2018/10/28/in-china-facial-recognition-tech-is-watching-you/
- 'Mission: Impossible - Rogue Nation', 2015.
- http://www.rapdataset.com/rapv1.html

Motivation: Person verification

This task is similar to other object re-identification

- Luggage
- Tiger
- Car



Reference

- MVB: A Large-Scale Dataset for Baggage Re-Identification, https://arxiv.org/pdf/1907.11366.pdf, https://sites.google.com/view/wacv2020animalreid/home
- https://cvwc2019.github.io/challenge.html
- https://github.com/JDAI-CV/VeRidataset

Idea 1: Classification

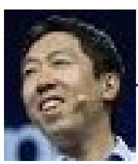
Multiple class classification

- Classification with a single label per sample
- Multiple classes (e.g., 1000+ samples per class)

Challenge

- What if we have insufficient data for certain label?
- What if we have 5 million classes, but < 10 samples per class?

Input image



Classifier (either traditional machine learning or deep learning methods) trained from image dataset with annotation

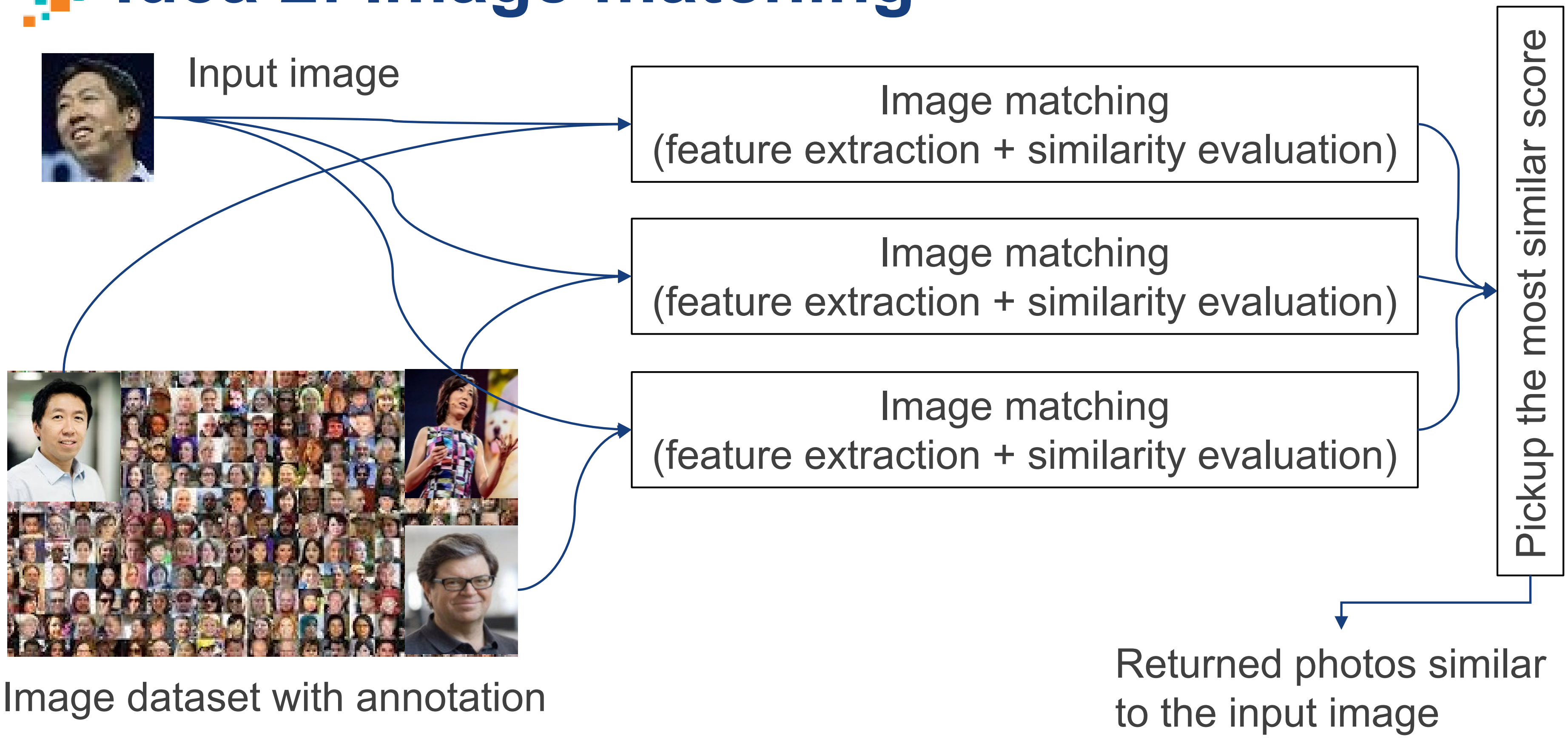
Andrew Ng

Li Fei-Fei

Yan Lecunn

Image dataset with annotation

Idea 2: Image matching



The traditional approach for matching images, relies on the following pipeline:

- Feature extraction, e.g., color histograms, LBP, HoG, pre-trained CNN.
- Similarity evaluation, e.g., Euclidean distance (see next slide).

Feature similarity evaluation

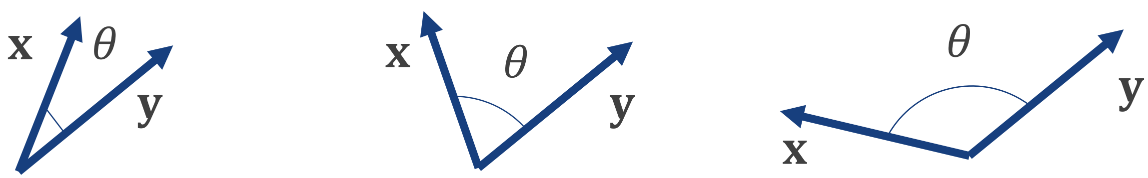
For unit vectors $\mathbf{x}, \mathbf{y}$, we have various pre-defined metrics, which are fully specified without the knowledge of data.

- Euclidian distance: $f(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 = (\mathbf{x} - \mathbf{y})^T\ (\mathbf{x} - \mathbf{y})$
- Cosine similarity distance: $f(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n}(x_i^2)}\sqrt{\sum_{i=1}^{n}(y_i^2)}}$

Dot product

Euclidean norm (i.e., length of vector)

Scenario	Similar	Unrelated	Opposite
Input two vectors $\mathbf{x}, \mathbf{y}$	Same direction	Nearly orthogonal	Opposite direction
Angle between them θ	Near 0 degree	Near 90 degree	Near 180 degree
Similarity score	Near 1	Near 0	Near -1



Reference: M. P. Chandra, On the generalised distance in statistics. *Proc. of the National Institute of Sciences of India*, Vol. 2, No. 1, 1936, pp. 49-55.
http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

Problem statement

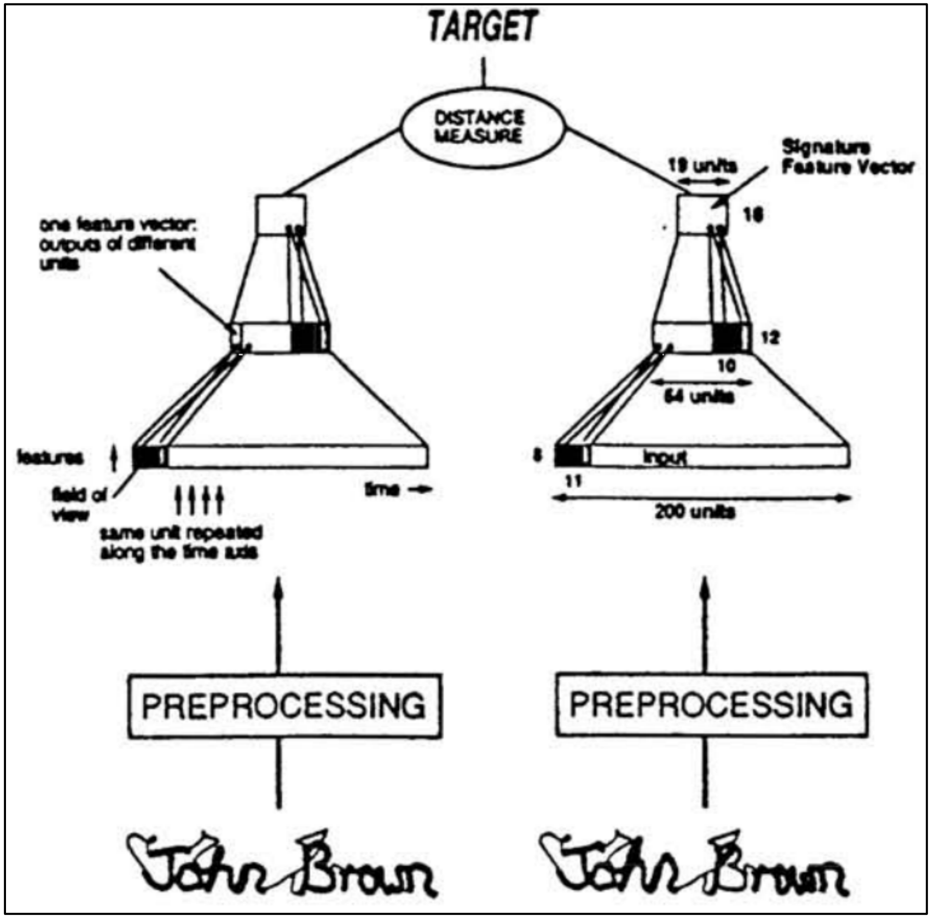
- Challenge in traditional image matching approach: The feature representation and the similarity metric are not learned jointly.
- Idea: A new problem statement
- Input: Given a pair of input images, we need to evaluate how “similar” they are to each other.
- Output: Either a binary label, i.e., 0 (same) or 1 (different), or a real number indicating how similar a pair of images are.

Images: Labeled Faces in the Wild, http://vis-www.cs.umass.edu/lfw/

Siamese network: Idea

Siamese neural network is a class of neural network architectures that contain two or more identical subnetworks, which have the same configuration with the same parameters and weights.

- Sharing weights across subnetworks means fewer parameters.
- Each subnetwork essentially produces a representation of its input. If your inputs are matching two pictures, it makes sense to use similar model to process similar inputs. This way you have representation vectors with the same semantics.

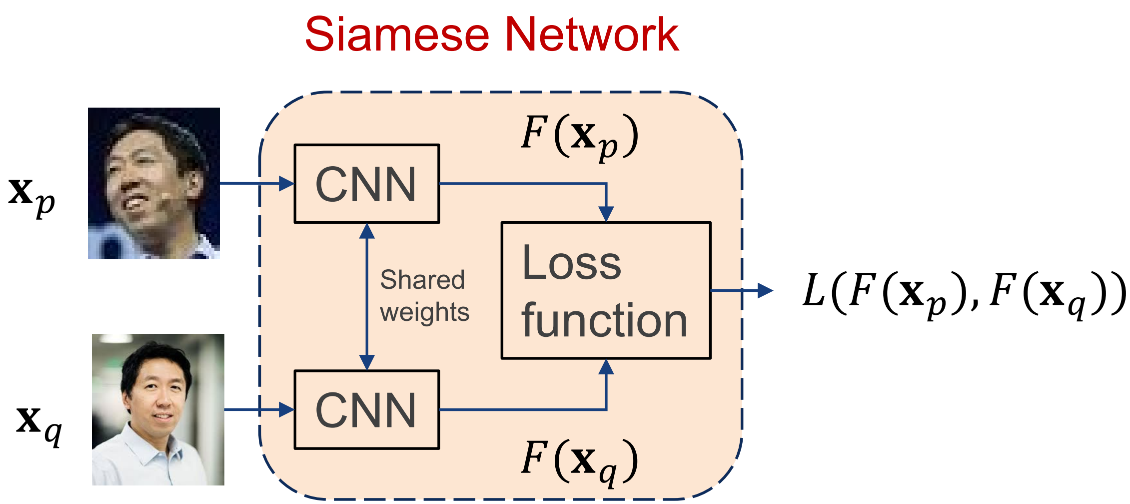


Reference: Bromley, et al., Signature verification using a Siamese time delay neural network, NIPS 1993, https://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf

Siamese CNN: Standard architecture

Siamese network

- Sample <u>positive</u> pairs $(\mathbf{x}_p, \mathbf{x}_q)$, with (p, q) of <u>same</u> class.
- Sample <u>negative</u> pairs $(\mathbf{x}_p, \mathbf{x}_q)$, with (p, q) of <u>different</u> classes.
- Forward pass using both inputs through the two networks (sharable weights). Back propagate through the two networks (the weights are updated with the sum of the two gradients).



Reference

- J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger and R. Shah, Signature Verification Using A "Siamese" Time Delay Neural Network. *IJPRAI*, Vol. *7, No.* 4, 1993, pp.669-688.
- E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, Discriminative learning of deep convolutional feature point descriptors, ICCV 2015.

Siamese CNN: Loss function

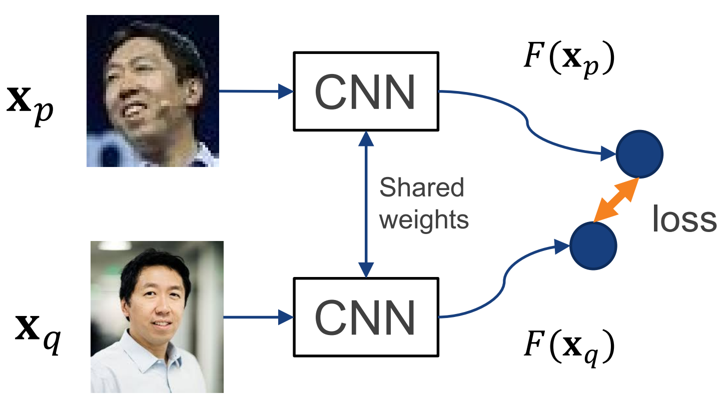
<u>Contrastive Loss</u> (y is a binary label, $y = 1$ for positive, $y = 0$ for negative)

$$L\big(F(\mathbf{x}_p), F(\mathbf{x}_q), y\big) = y * ||F(\mathbf{x}_p) - F(\mathbf{x}_q)||^2 + (1 - y) * \max\big(0, m^2 - ||F(\mathbf{x}_p) - F(\mathbf{x}_q)||^2\big)$$

Positive pair loss (Euclidian Loss):

$$L(F(\mathbf{x}_p), F(\mathbf{x}_q)) = ||F(\mathbf{x}_p) - F(\mathbf{x}_q)||^2$$

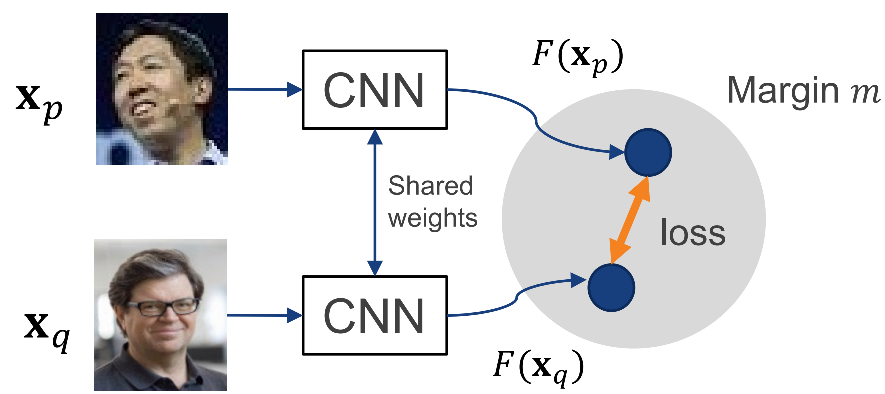
If model is good	$\lVert F(\mathbf{x}_p) - F(\mathbf{x}_q)\rVert^2$ is small	Loss is small	Model will be slightly updated
If model is bad	$\lVert F(\mathbf{x}_p) - F(\mathbf{x}_q)\rVert^2$ is large	Loss is large	Model will be updated



Negative pair loss (given a small user-defined margin m, Hinge Loss):

$$L(F(\mathbf{x}_p), F(\mathbf{x}_q)) = \max\big(0, m^2 - ||F(\mathbf{x}_p) - F(\mathbf{x}_q)||^2\big)$$

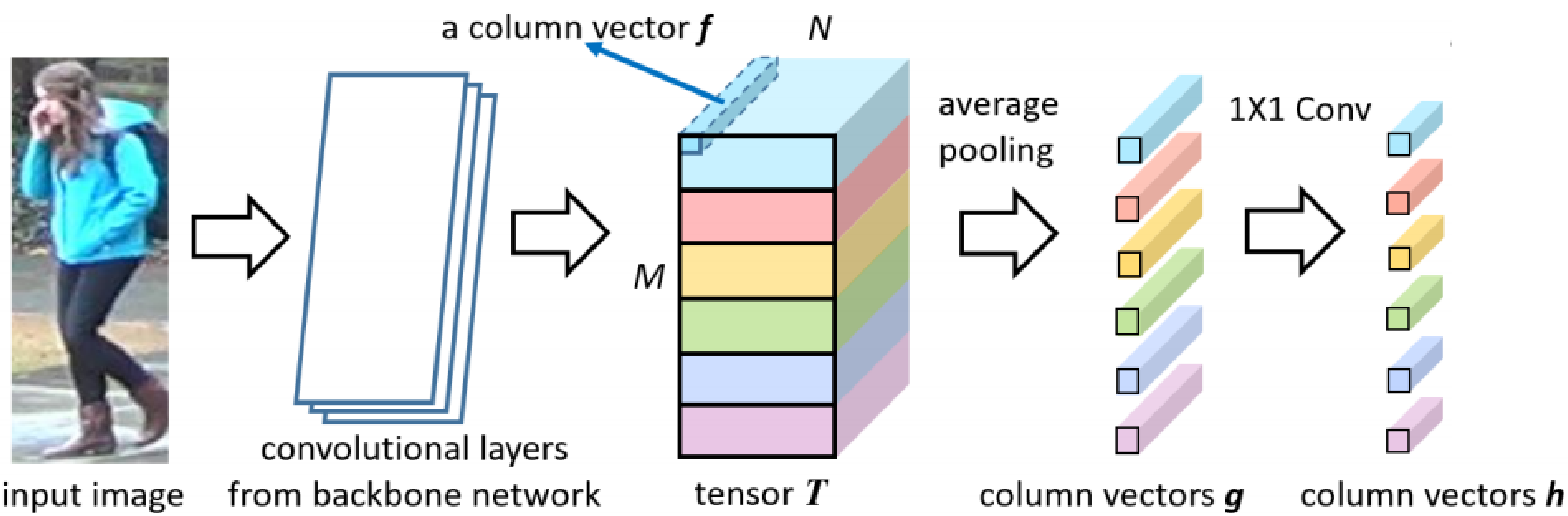
If model is good	$\lVert F(\mathbf{x}_p) - F(\mathbf{x}_q)\rVert^2$ is large positive	Loss is 0	Model will not be updated
If model is bad	$\lVert F(\mathbf{x}_p) - F(\mathbf{x}_q)\rVert^2$ is small positive	Loss is small	Model will be updated



Reference: S. Bell and K. Bala, Learning visual similarity for product design with convolutional neural networks. *ACM Trans. on Graphics*, Vol. 34, No. 4, 2015, *https://www.cs.cornell.edu/~kb/publications/SIG15ProductNet.pdf*

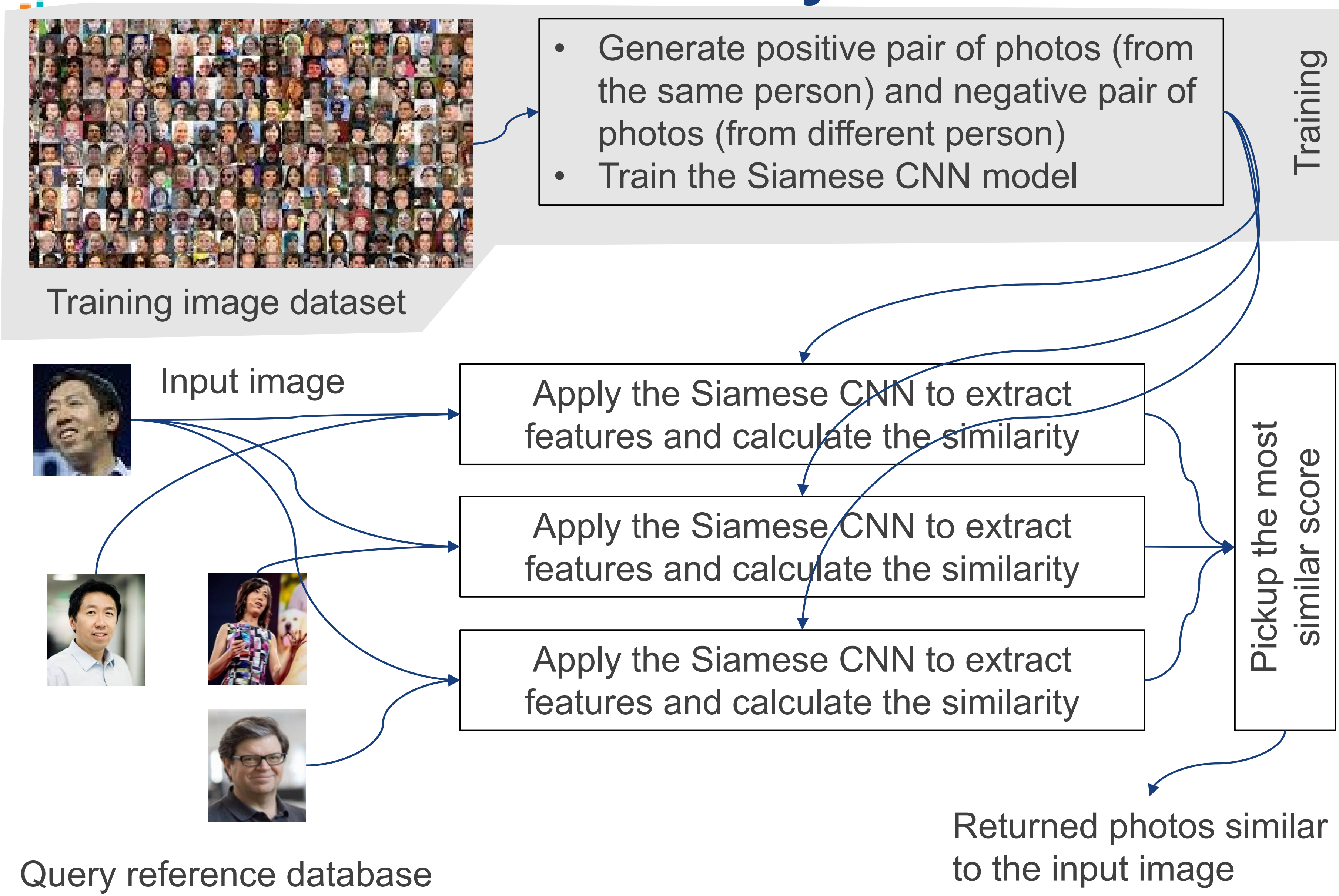
Further consider local features

Idea: The input image goes forward through the backbone CNN network to form a 3D tensor $\boldsymbol{T}$. Then, it replaces the original global pooling layer to spatially down-sample $\boldsymbol{T}$ into a few pieces of column vectors $\boldsymbol{g}$. A following 1×1 convolutional layer reduces the dimension of $\boldsymbol{g}$ to obtain dimension-reduced column vector $\boldsymbol{h}$, which are concatenated to form the final descriptor (to be used in the loss function studied in the previous slides) of the input image.

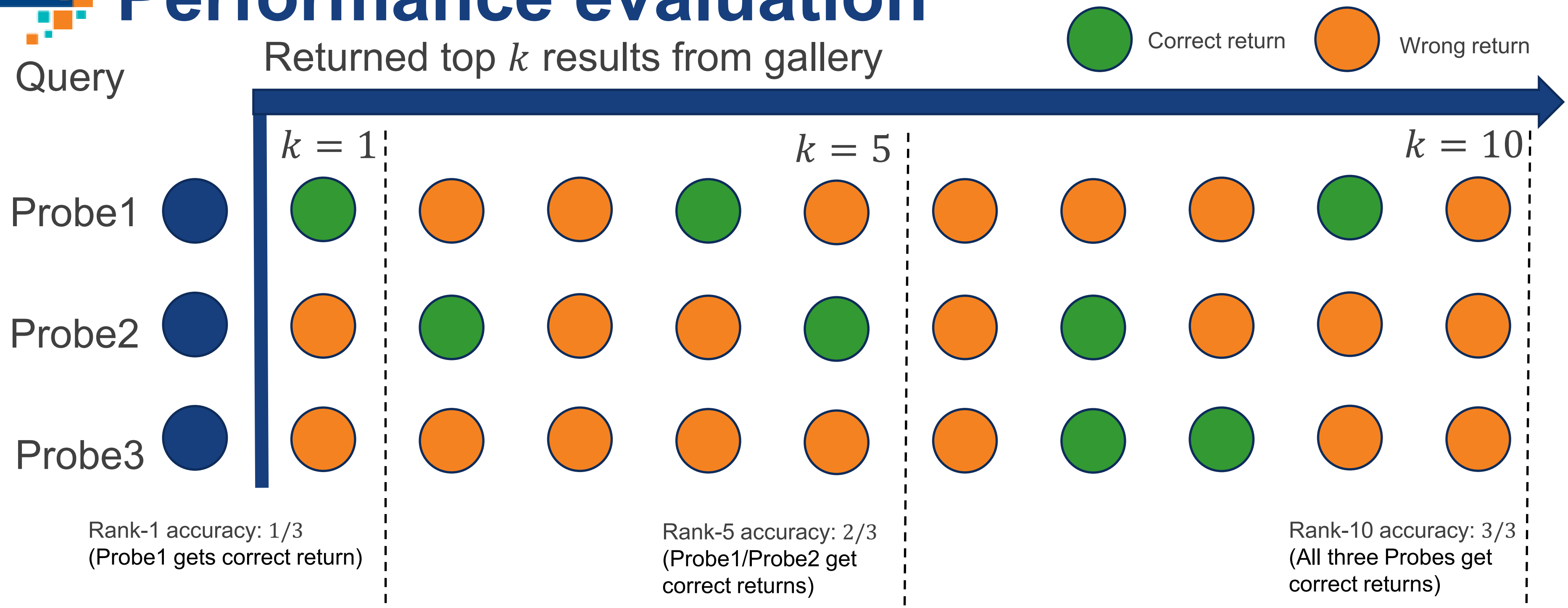


Reference: Beyond Part Models: Person Retrieval with Refined Part Pooling (and a Strong Convolutional Baseline), https://arxiv.org/abs/1711.09349

Siamese CNN: Summary



Performance evaluation



- Query: Given probes (e.g., 3 images) of the target person, evaluate each image in the dataset (gallery), rank the returned results in terms of the similarity scores.
- Rank-k accuracy: Whether the system gets correctly (at least one) in the top k returned results.
- mAP (*mean average precision*): Average precision for all probes.
  - Probe1: $AP = (1/1 + 2/4 + 3/9)/3 = 0.61$
  - Probe2: $AP = (1/2 + 2/5 + 3/7)/3 = 0.44$
  - Probe3: $AP = (1/7 + 2/8)/2 = 0.20$
  - Overall: $mAP = (0.61 + 0.44 + 0.2)/3 = 0.42$

Application: Person Re-Identification

True positive

True negative

Reference: E. Ahmed, M. Jones and T. K. Marks, An improved deep learning architecture for person re-identification, CVPR 2015, https://www.merl.com/publications/docs/TR2015-076.pdf

Workshop: Person Re-Identification

Dataset: *Labeled Faces in the Wild* (LFW) dataset, http://vis-www.cs.umass.edu/lfw/
- Evaluate similarity of two person image using HoG feature extraction method.
- Build a person verification model using Siamese deep learning method.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 100, 100, 3)       0
_________________________________________________________________
input_3 (InputLayer)         (None, 100, 100, 3)       0
_________________________________________________________________
model_1 (Model)              (None, 50)                2002818
_________________________________________________________________
dot_1 (Dot)                  (None, 1)                 0
=================================================================
Total params: 2,002,818
Trainable params: 2,002,818
Non-trainable params: 0
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 100, 100, 3)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 100, 100, 16)      448
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 100, 100, 16)      2320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 50, 50, 16)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 40000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                2000050
=================================================================
Total params: 2,002,818
Trainable params: 2,002,818
Non-trainable params: 0
```

$Conv2d(3,3)$

Parameter calculation

$$(3 \times 3 \times 3 + 1) \times 16 = 448$$
$$(3 \times 3 \times 16 + 1) \times 16 = 2320$$
$$(40000 + 1) \times 50 = 2000050$$

Generated pairs of positive/negative samples

What we have learnt

- Feature extraction using texture, gradients.
- Feature representation learning, Siamese network.
- Application: Person re-identification in surveillance

Thank you!

Dr TIAN Jing
Email: tianjing@nus.edu.sg

