Some Computer Vision Basics

Michelle Torres
Washington University in St. Louis

April 21, 2019

Image as Data: Motivation

- Images are powerful: extra information + emotional activation + *see to believe* = recall and engagement
- Images are (kind of) universal
- Visuals are frames
- New possibilities for measurement: ethnic visibility, natural disaster prevention and evaluation, state capacity, data collection, etc.
- **Computer Vision:**
  - Teaching computers to see
  - Process large pools of images
  - Increase consistency/reliability and decrease bias (*)
  - Helping humans to "see" and discover (*)

Image as Data: Teaching the computer to see

Image as Data: Teaching the computer to see

A very hard task!

Image as Data: Teaching the computer to see

A very hard task!

- Computers are great at following instructions reliably...

Image as Data: Teaching the computer to see

A very hard task!

- Computers are great at following instructions reliably...
- ... but they are bad at inferences

Image as Data: Teaching the computer to see

A very hard task!

- Computers are great at following instructions reliably...
- ... but they are bad at inferences

Image as Data: Teaching the computer to see

A very hard task!

- Computers are great at following instructions reliably...
- ... but they are bad at inferences

Image as Data: Teaching the computer to see

A very hard task!

- Computers are great at following instructions reliably...
- ... but they are bad at inferences

Overview

1. Image Basics
   - Pixels, color channels, and more
   - Software and computational requirements
2. Extracting basic features
   - Color statistics
3. The Bag of Visual Words
4. Convolutional Neural Networks
   - Overview
   - Structure and implementation
   - Application
5. Miscellaneous
   - GoogleVision API
   - Validation and diagnosis
   - Warnings: scope, biases and unintended problems

GETTING READY

- Workshop material available at `https://github.com/smtorres/StanfordClass`
- Install Keras (`https://keras.io/#installation`)
- Install the following python libraries: numpy, scipy, cv2, matplotlib, PIL, sklearn
  - Check tutorials for OpenCV installation: `https://www.pyimagesearch.com/opencv-tutorials-resources-guides/`
  - I suggest OpenCV 3.X and its compilation from source for full funcionality

IMAGE BASICS

- An image is a set of pixels:

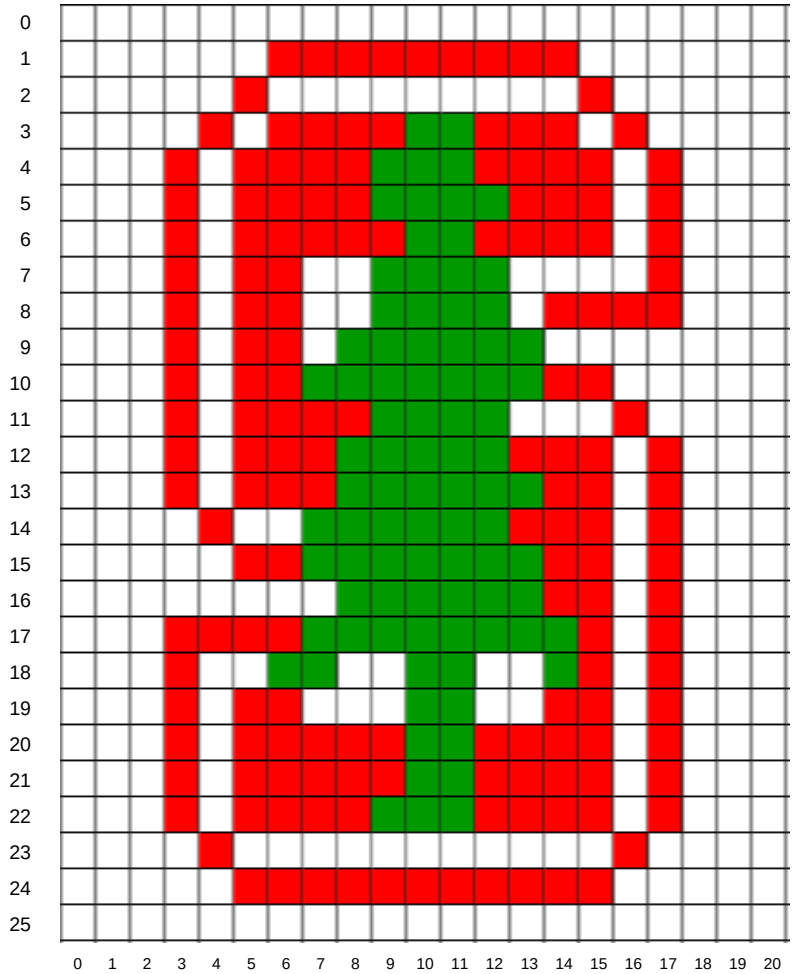


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)

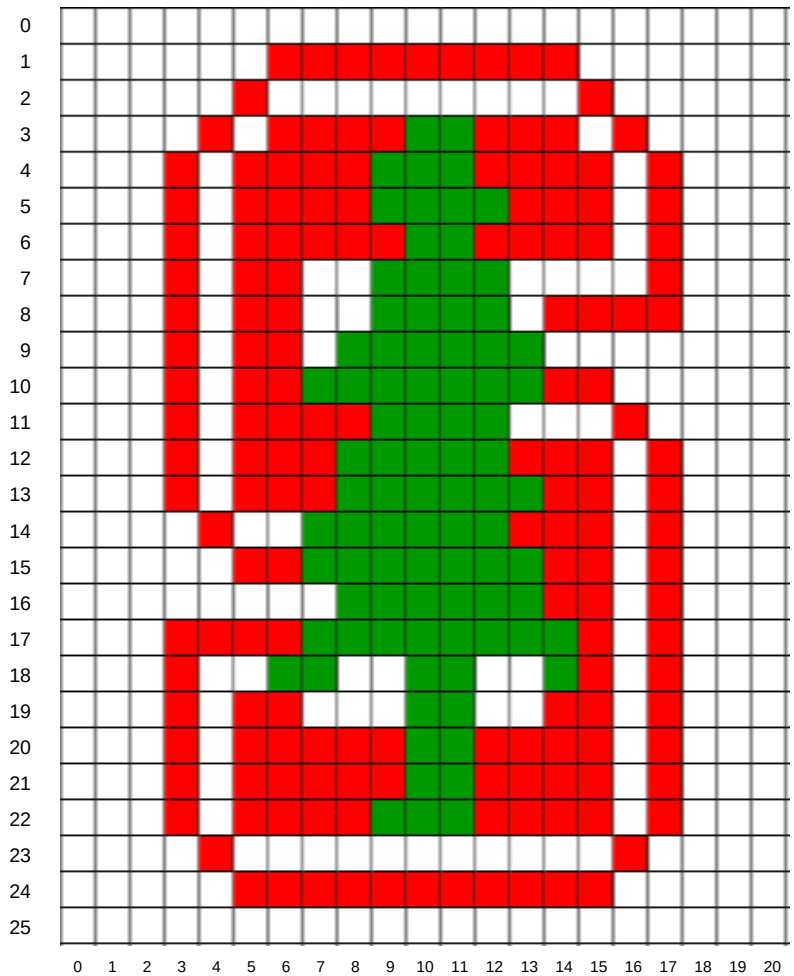


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)
  - Grayscale: intensity of light, Color: color intensity.

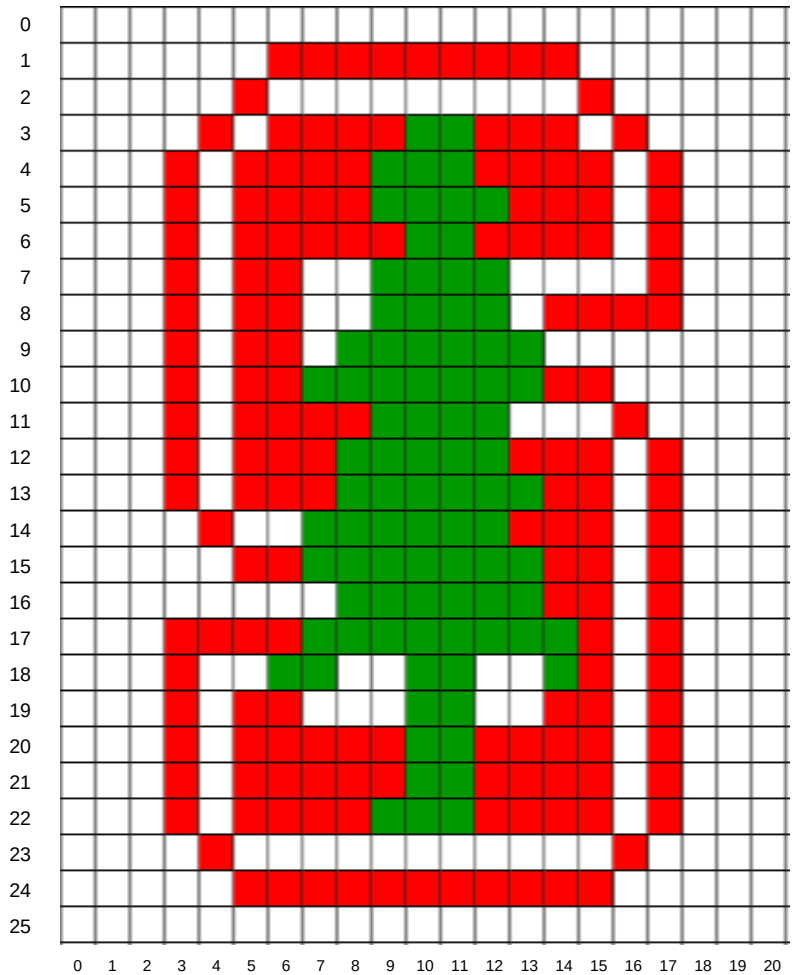


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)
  - Grayscale: intensity of light, Color: color intensity.
- Matrix representation

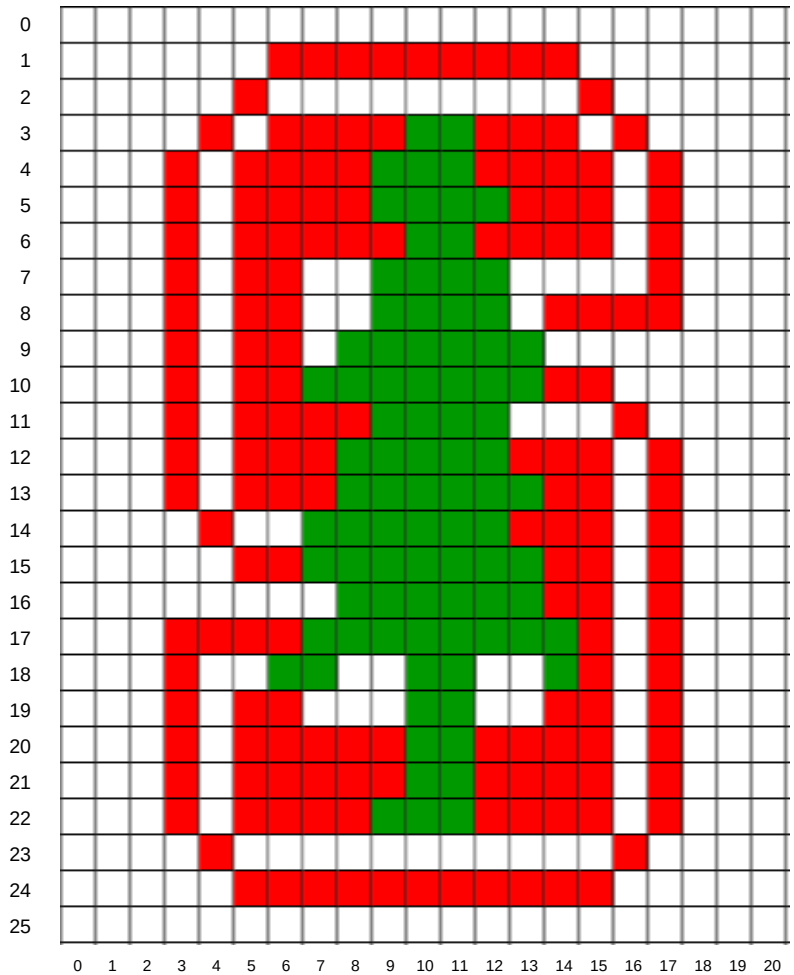


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)
  - Grayscale: intensity of light, Color: color intensity.
- Matrix representation
  - Grayscale: one matrix

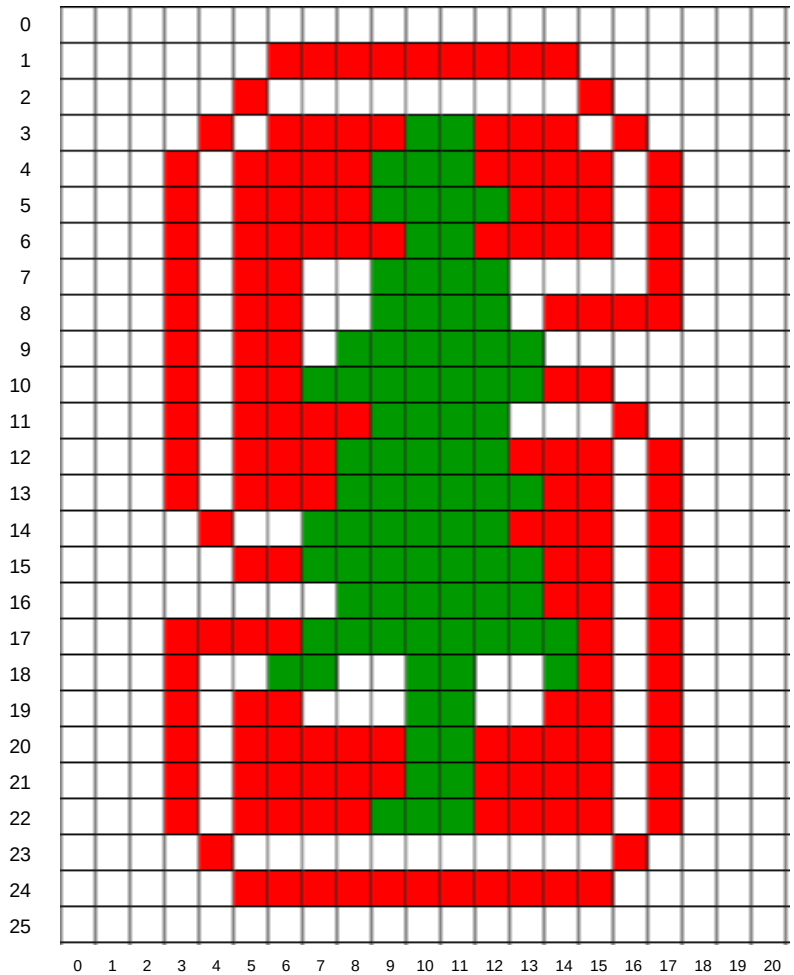


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)
  - Grayscale: intensity of light, Color: color intensity.
- Matrix representation
  - Grayscale: one matrix
  - Color: array with a matrix for each color channel (Red, Green, and Blue)

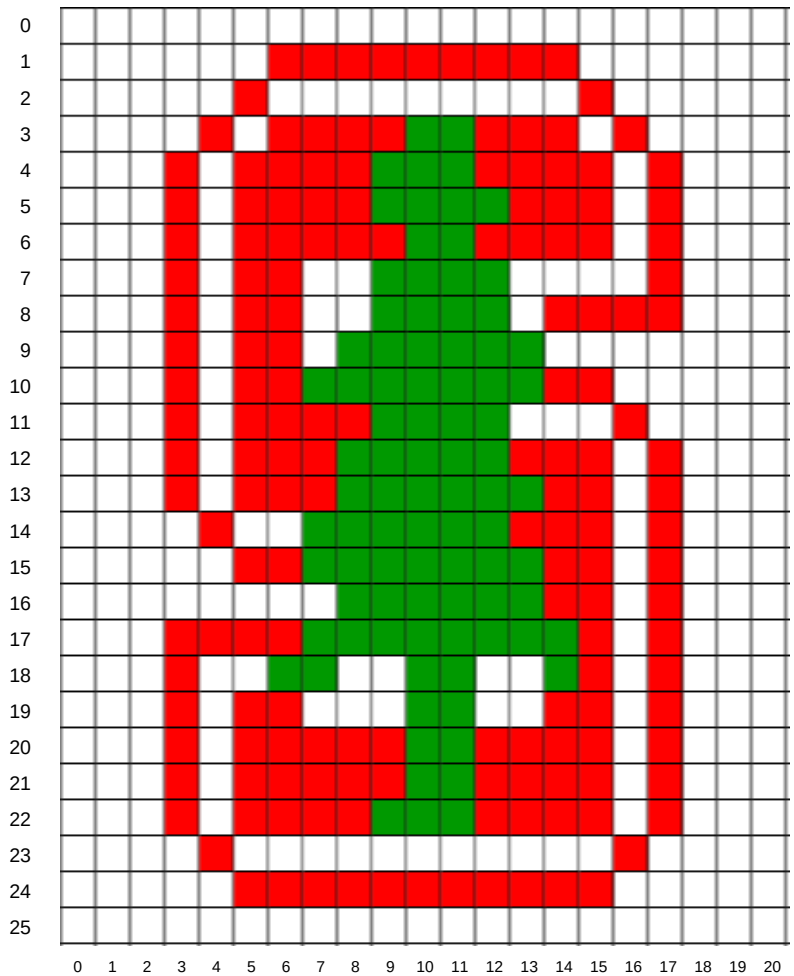


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)
  - Grayscale: intensity of light, Color: color intensity.
- Matrix representation
  - Grayscale: one matrix
  - Color: array with a matrix for each color channel (Red, Green, and Blue)
- Notice that in OpenCV:

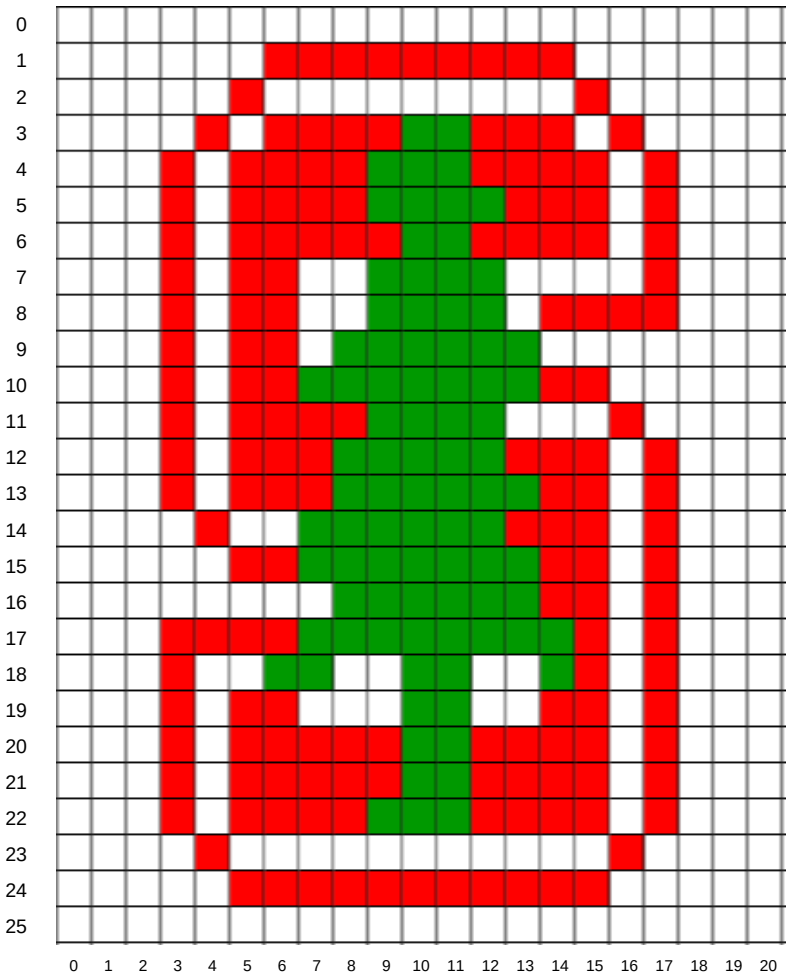


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)
  - Grayscale: intensity of light, Color: color intensity.
- Matrix representation
  - Grayscale: one matrix
  - Color: array with a matrix for each color channel (Red, Green, and Blue)
- Notice that in OpenCV:
  - Color channel specification is BRG instead of RGB

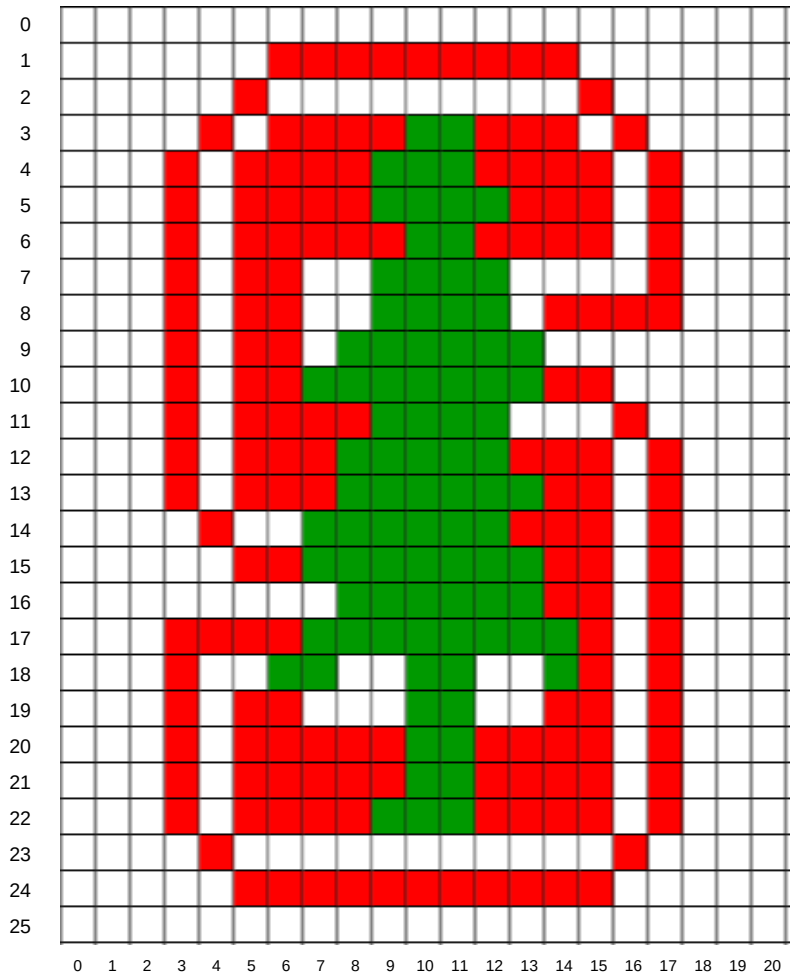


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)
  - Grayscale: intensity of light, Color: color intensity.
- Matrix representation
  - Grayscale: one matrix
  - Color: array with a matrix for each color channel (Red, Green, and Blue)
- Notice that in OpenCV:
  - Color channel specification is BRG instead of RGB
  - Origin of image is different (top left corner)

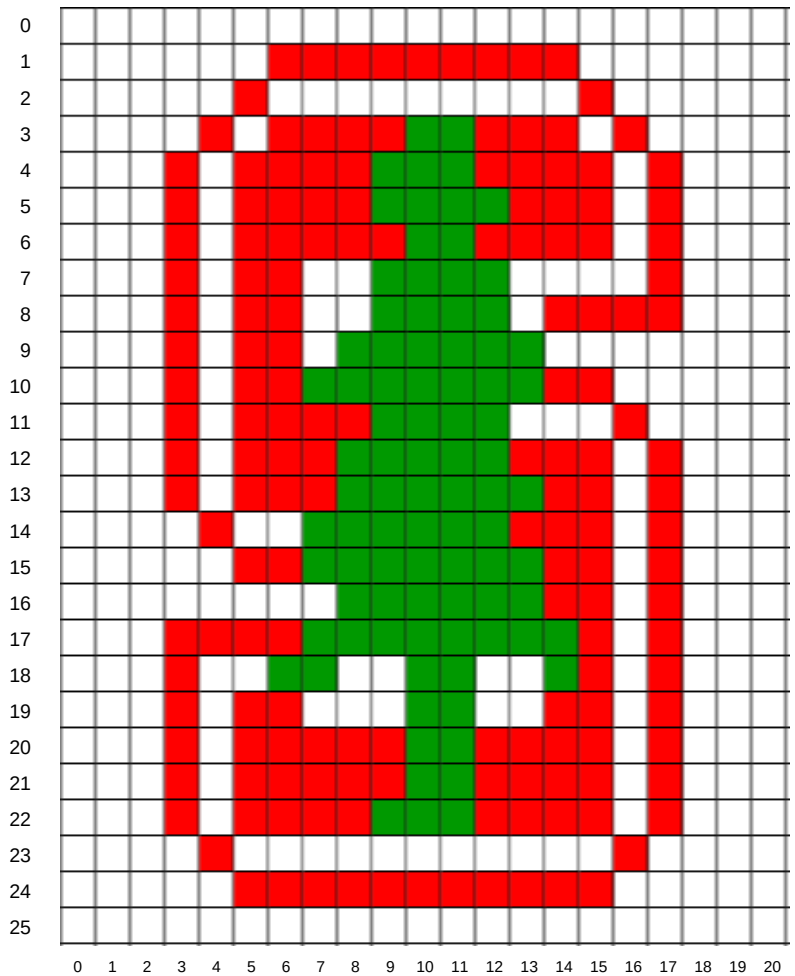
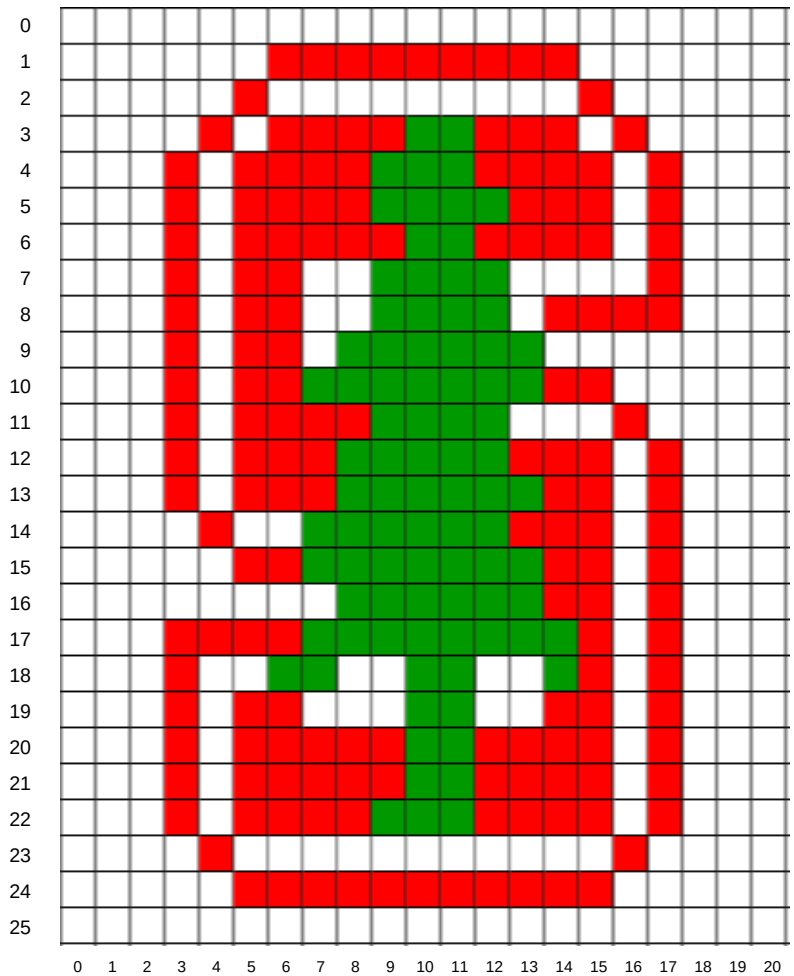


IMAGE BASICS

- An image is a set of pixels:
  - Finest unit (defines height and width)
  - Grayscale: intensity of light, Color: color intensity.
- Matrix representation
  - Grayscale: one matrix
  - Color: array with a matrix for each color channel (Red, Green, and Blue)
- Notice that in OpenCV:
  - Color channel specification is BRG instead of RGB
  - Origin of image is different (top left corner)
  - In `numpy` you specify the *y*–coordinates of an image first: `x2 = image[y0:y1, x0:x1]`



Color descriptors

Describing an image

- A challenge: a lot of pixels that mostly make sense when analyzed in clusters and not as units.
- Therefore, we use **image descriptors** to characterize the content on an image ***globally*** or feature descriptors to locally quantify ***regions*** of the image.
  - Color
  - Texture
  - Shape
- Feature vectors: A series of numbers used to numerically quantify the contents of an image (or regions of it)

Color statistics

Channel statistics

- Very intuitive and simple
- Basic statistics of each color channel
  1. Separate channels
  2. Compute moments for each channel

Histograms

- More information based on distribution
- 3D histogram of colors
  1. Convert image to L*a*b color space
  2. Compute 3D histogram

4. Concatenate to form *feature vector*
5. Compute vector similarity

Color statistics

Channel statistics

- Very intuitive and simple
- Basic statistics of each color channel
  1. Separate channels
  2. Compute moments for each channel

Histograms

- More information based on distribution
- 3D histogram of colors
  1. Convert image to L*a*b color space
  2. Compute 3D histogram

4. Concatenate to form *feature vector*
5. Compute vector similarity

$$d(p,q) = \sqrt{\sum_{i=1}^{N}(q_i - p_i)^2}$$

COLOR STATISTICS, CONT.

The Bag of Visual Words

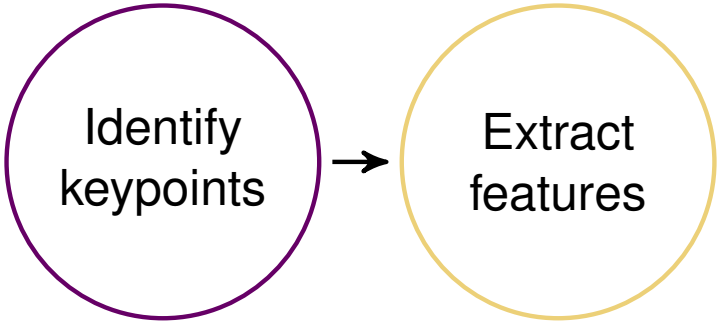
The BoVW

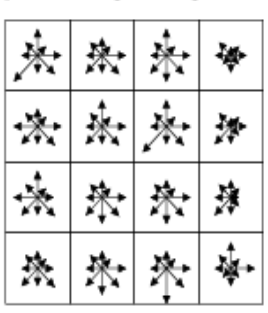
- Emulate a Document-Term matrix
- No words *but* patches

Identify keypoints

The BoVW

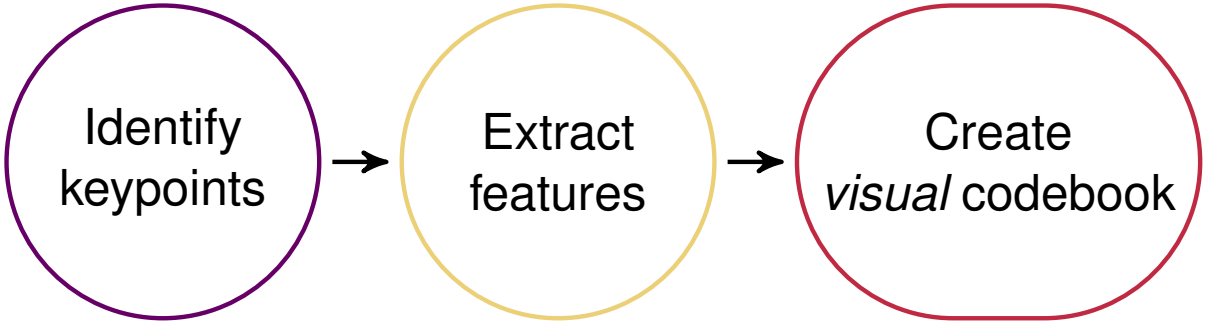
- Emulate a Document-Term matrix
- No words *but* patches

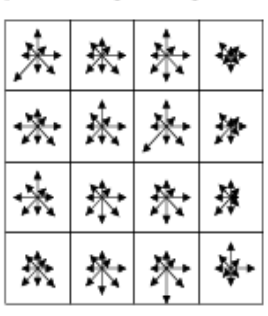


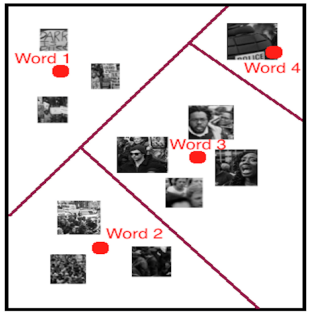


The BoVW

- Emulate a Document-Term matrix
- No words *but* patches

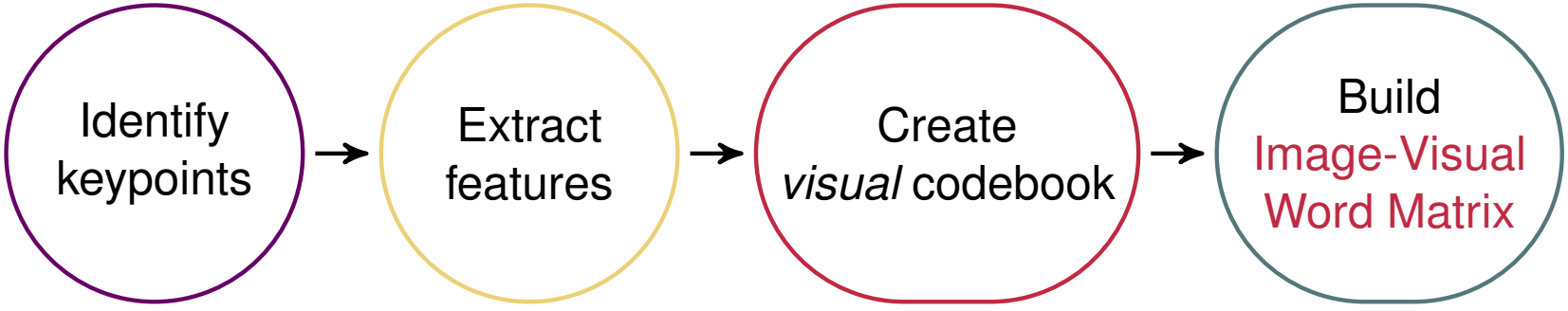


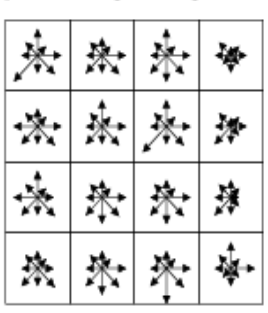


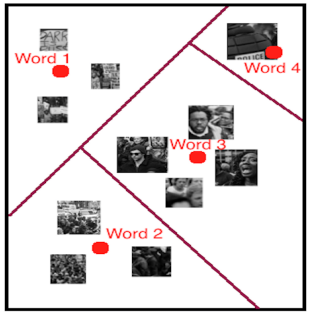


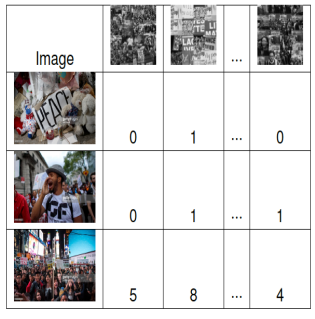
The BoVW

- Emulate a Document-Term matrix
- No words *but* patches









KEY POINT DETECTION

- Identify salient regions, edges and corners based on pixel intensity changes
- FAST Hessian: fast and scale-invariant (Bay et al. 2006)

KEY POINT DETECTION: HOW TO IDENTIFY THEM?

- FAST Hessian: fast and scale-invariant (Bay et al. 2006)
  1. Scale (smooth) images to create *octaves*
     - Convolution with an approximation of a Gaussian second-order derivatives filter

KEY POINT DETECTION: HOW TO IDENTIFY THEM?

- FAST Hessian: fast and scale-invariant (Bay et al. 2006)
  1. Scale (smooth) images to create *octaves*
     - Convolution with an approximation of a Gaussian second-order derivatives filter

$$L_{xx} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial x^2} g(\sigma) \quad L_{yy} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial y^2} g(\sigma) \quad L_{xy} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial x \partial y} g(\sigma)$$

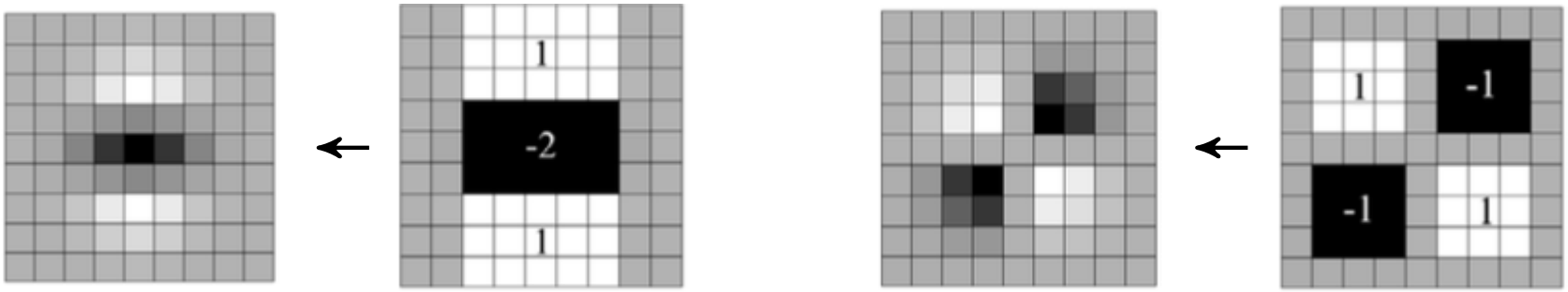
$$\mathcal{H}(\boldsymbol{x}, \sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x}, \sigma) & L_{xy}(\boldsymbol{x}, \sigma) \\ L_{xy}(\boldsymbol{x}, \sigma) & L_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix}$$

KEY POINT DETECTION: HOW TO IDENTIFY THEM?

- FAST Hessian: fast and scale-invariant (Bay et al. 2006)
  1. Scale (smooth) images to create *octaves*
     - Convolution with an approximation of a Gaussian second-order derivatives filter

$$L_{xx} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial x^2} g(\sigma) \quad L_{yy} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial y^2} g(\sigma) \quad L_{xy} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial x \partial y} g(\sigma)$$

$$\mathcal{H}(\boldsymbol{x}, \sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x}, \sigma) & L_{xy}(\boldsymbol{x}, \sigma) \\ L_{xy}(\boldsymbol{x}, \sigma) & L_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix}$$

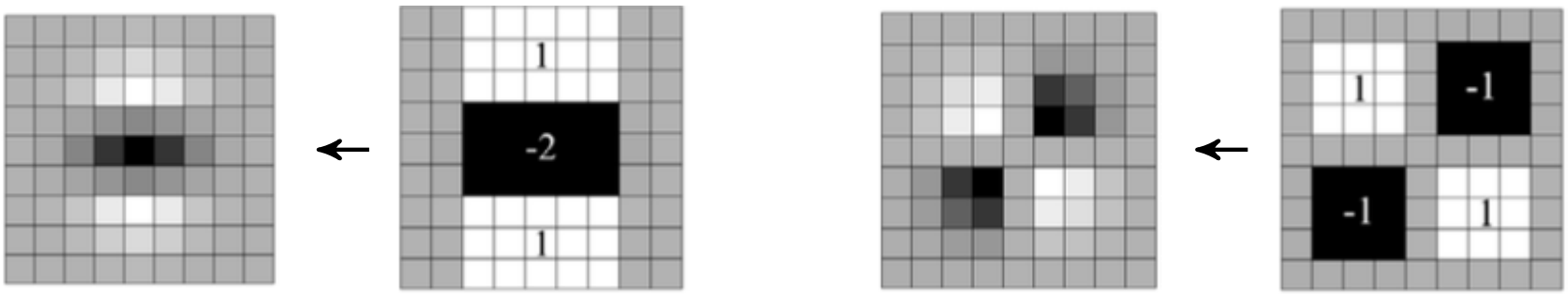


KEY POINT DETECTION: HOW TO IDENTIFY THEM?

- FAST Hessian: fast and scale-invariant (Bay et al. 2006)
  1. Scale (smooth) images to create *octaves*
     - Convolution with an approximation of a Gaussian second-order derivatives filter

$$L_{xx} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial x^2} g(\sigma) \quad L_{yy} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial y^2} g(\sigma) \quad L_{xy} = I(\boldsymbol{x}) \circledast \frac{\partial^2}{\partial x \partial y} g(\sigma)$$

$$\mathcal{H}(\boldsymbol{x}, \sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x}, \sigma) & L_{xy}(\boldsymbol{x}, \sigma) \\ L_{xy}(\boldsymbol{x}, \sigma) & L_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix}$$

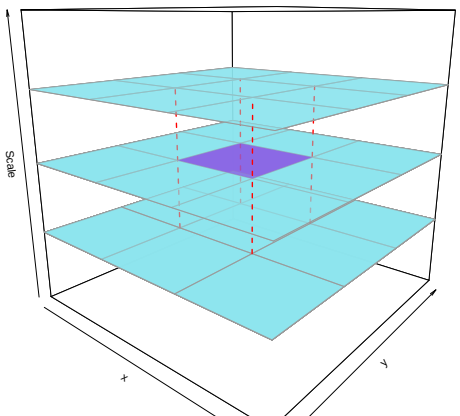


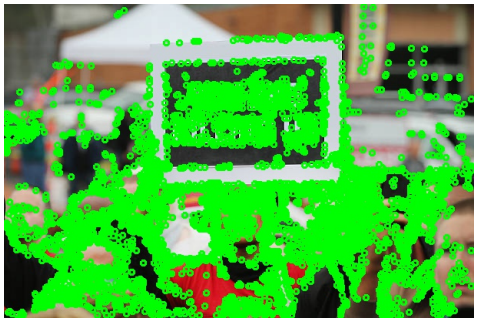
$$\text{Approximation} \Rightarrow \det(\mathcal{H}_{approx}) = D_{xx} D_{yy} - (0.9 D_{xy})^2$$

KEY POINT DETECTION: HOW TO IDENTIFY THEM?

KEY POINT DETECTION: HOW TO IDENTIFY THEM?

2. Compare pixels to their $3 \times 3 \times 3$ neighborhood





Feature extraction

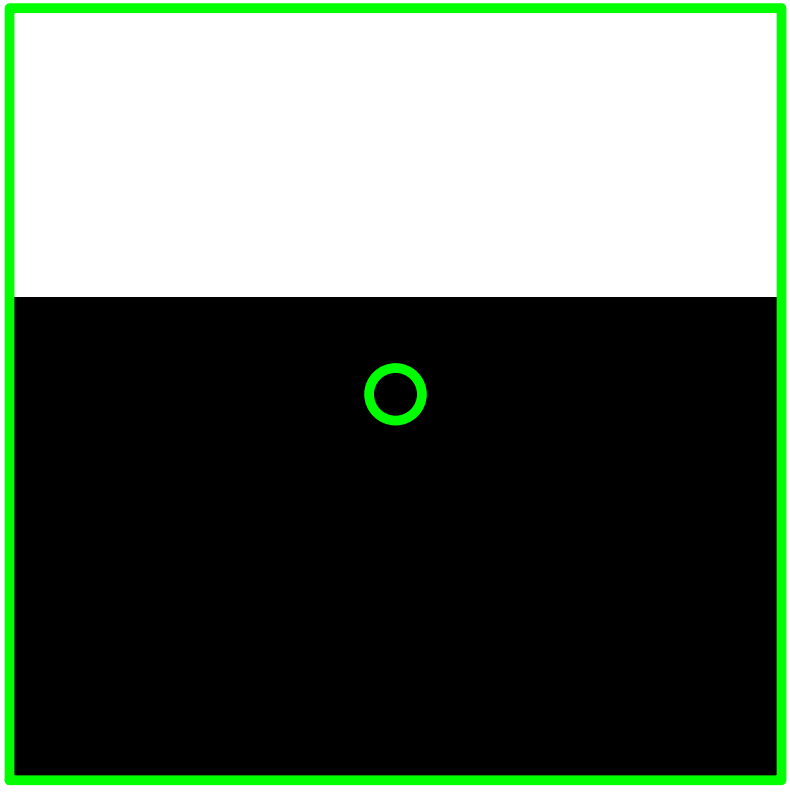
1. Gaussian blurred image (A)
2. 16×16 neighborhood around each key point, broken into 4×4 cells
3. 8-bin histogram of gradients' orientations weighted by magnitude, per cell
   - $G_x = A(x, y) - A(x+1, y), G_y = A(x, y) - A(x, y+1)$
   - $M_{x,y} = \sqrt{G_x^2 + G_y^2}$
   - $\theta_{x,y} = arctan2(G_y, G_x) \times \left(\frac{180}{\pi}\right)$
4. Square root and normalization

Feature vector

128-dimensional vector: 8-bins per histogram × 16 cells

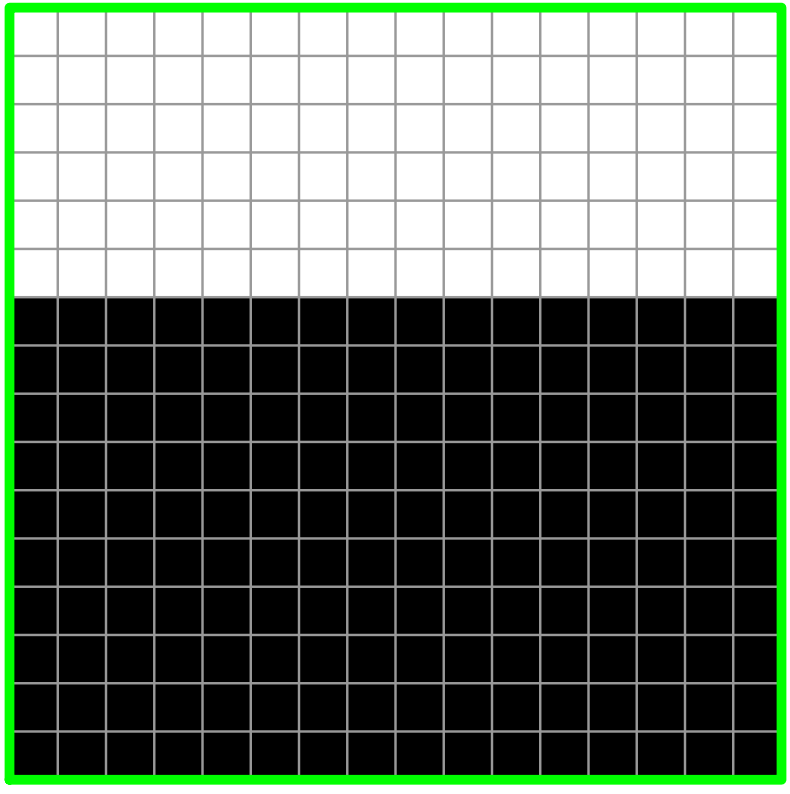
Feature identification: Feature extraction (cont.)

- Take a key point



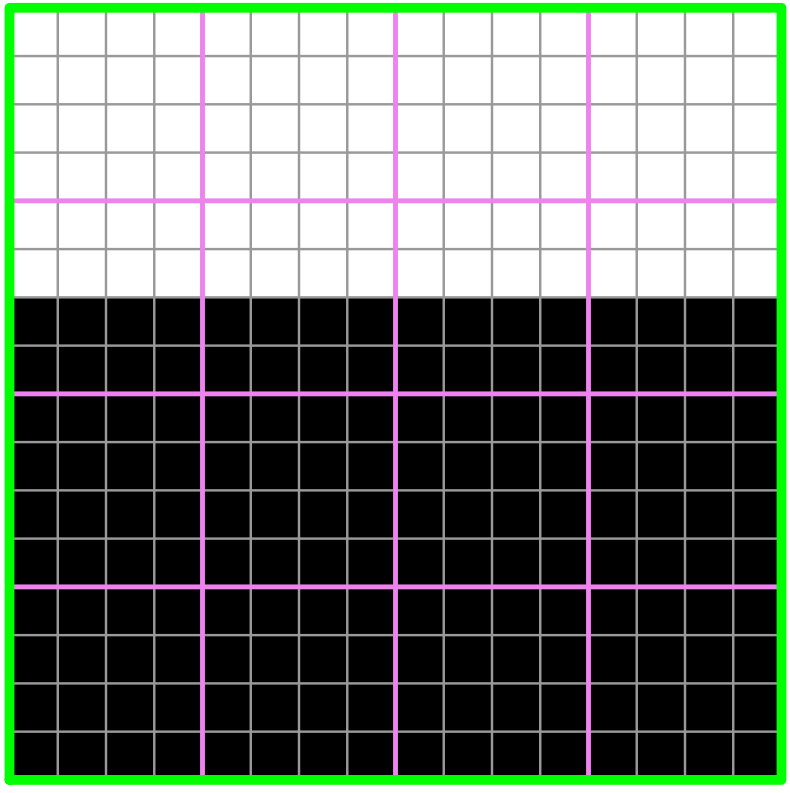
Feature identification: Feature extraction (cont.)

- Draw a 16 × 16 pixel neighborhood



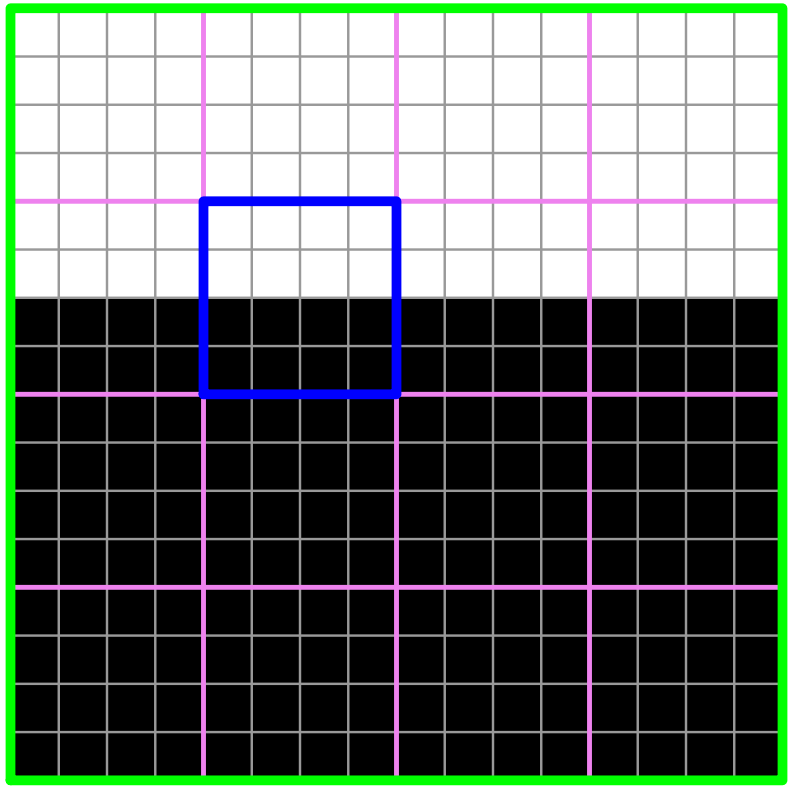
Feature identification: Feature extraction (cont.)

- Divide in 4 × 4 pixel cells



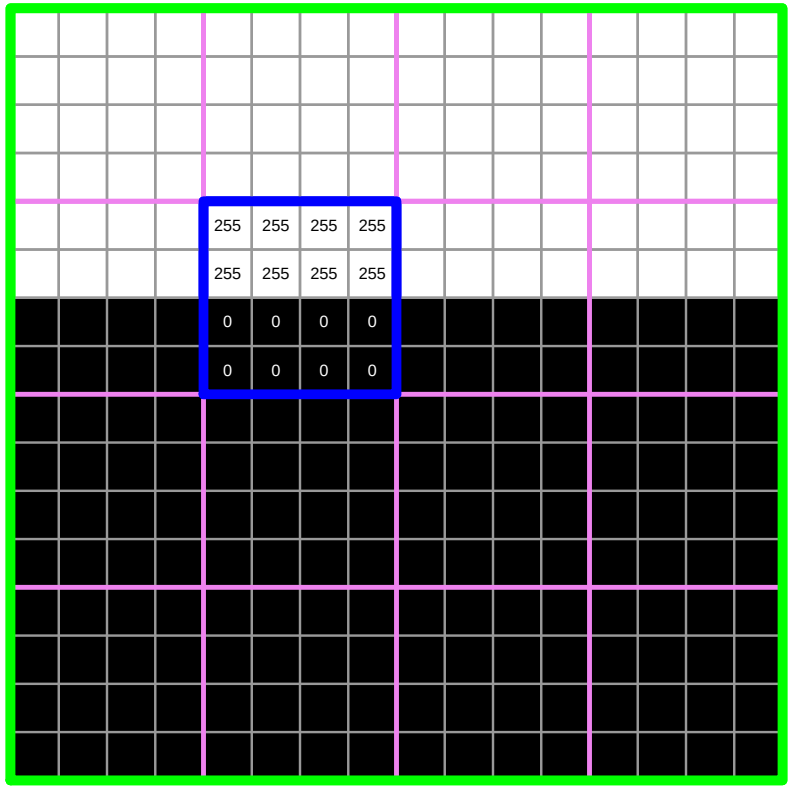
Feature identification: Feature extraction (cont.)

- Let's focus on one cell...



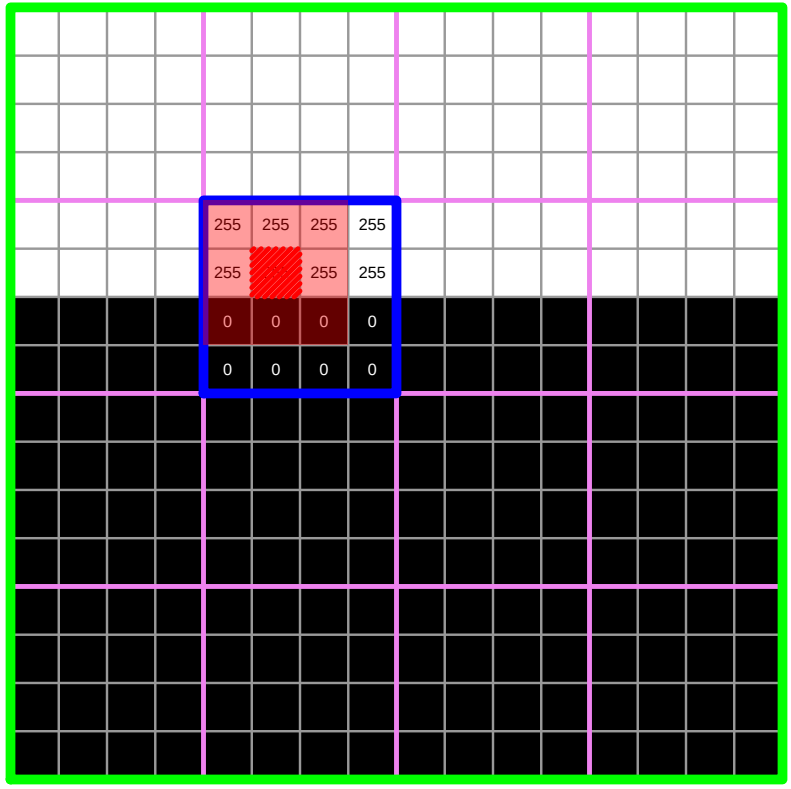
Feature identification: Feature extraction (cont.)

- Let's focus on one cell...



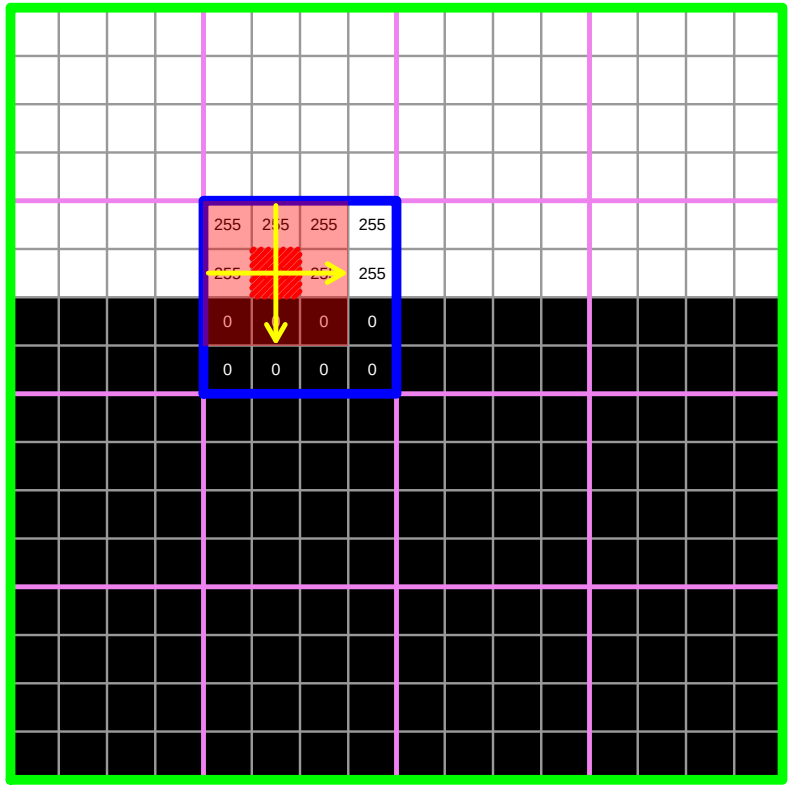
Feature identification: Feature extraction (cont.)

- Estimate gradient orientation and magnitude



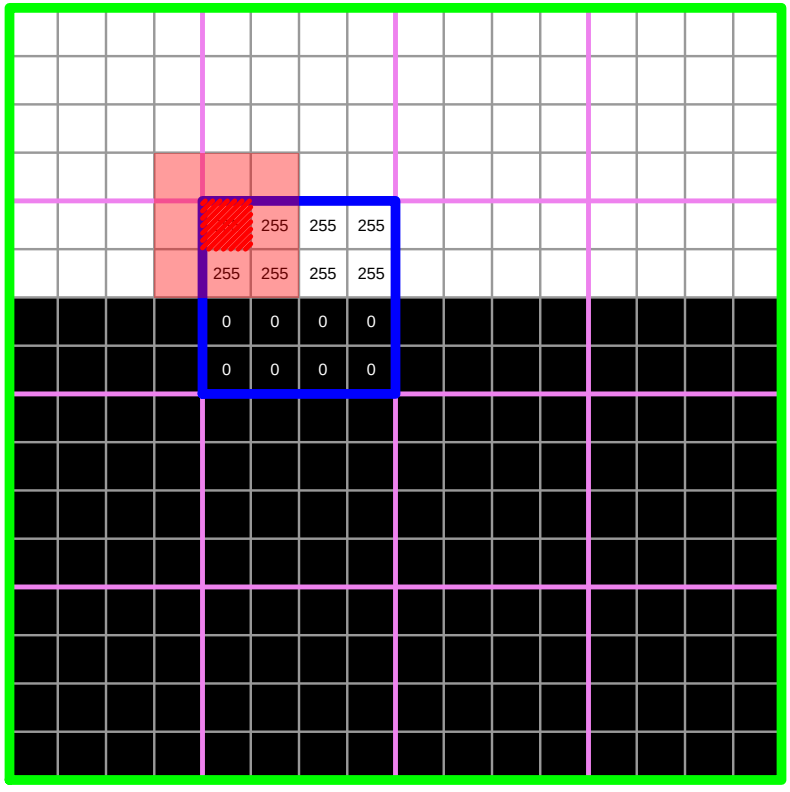
Feature identification: Feature extraction (cont.)

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad \theta = arctan2(G_x, G_y) * \frac{180}{\pi}$$



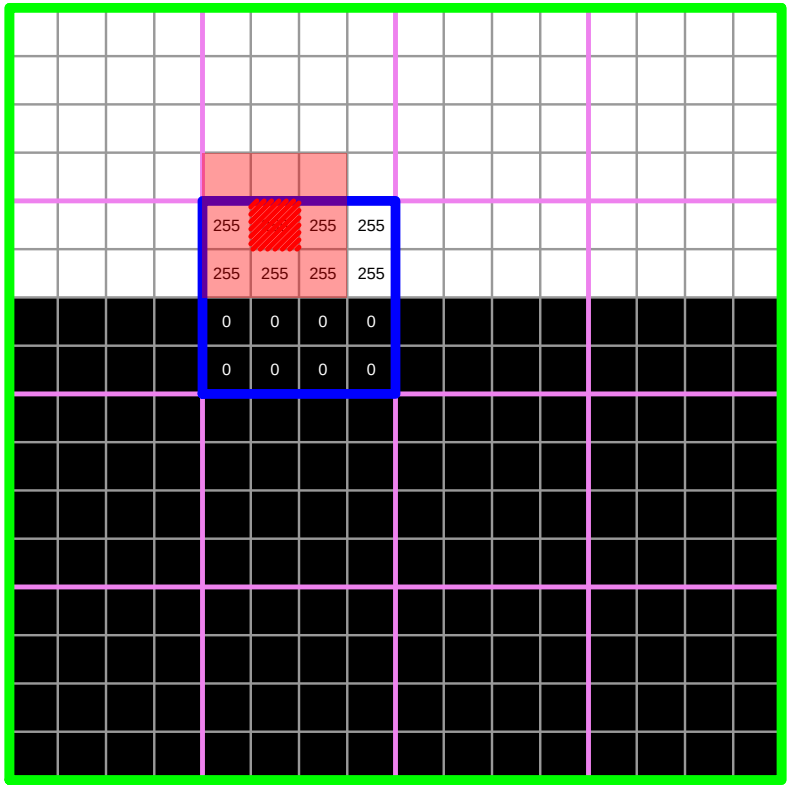
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



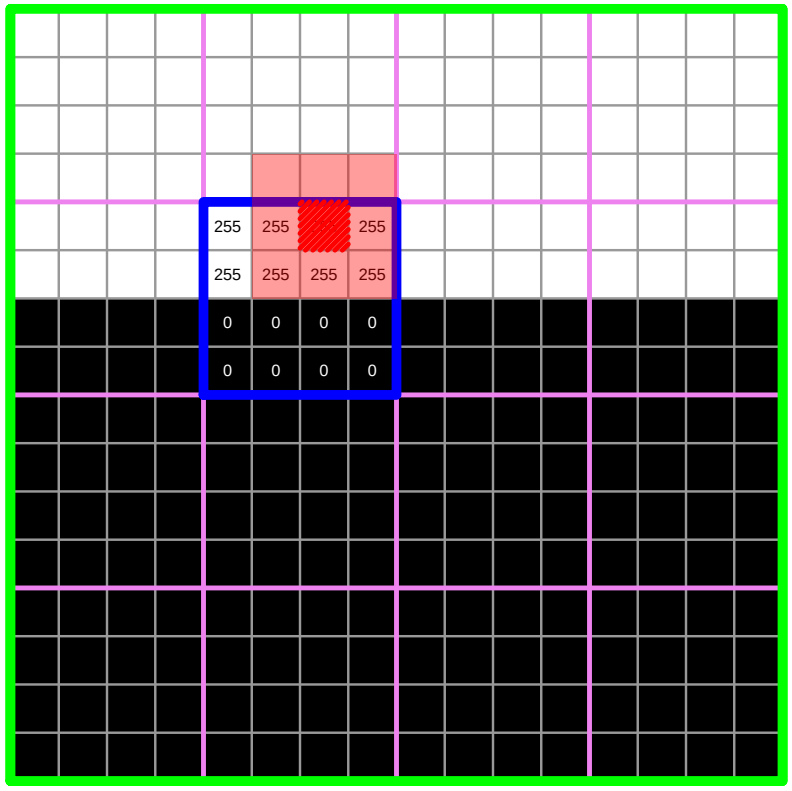
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



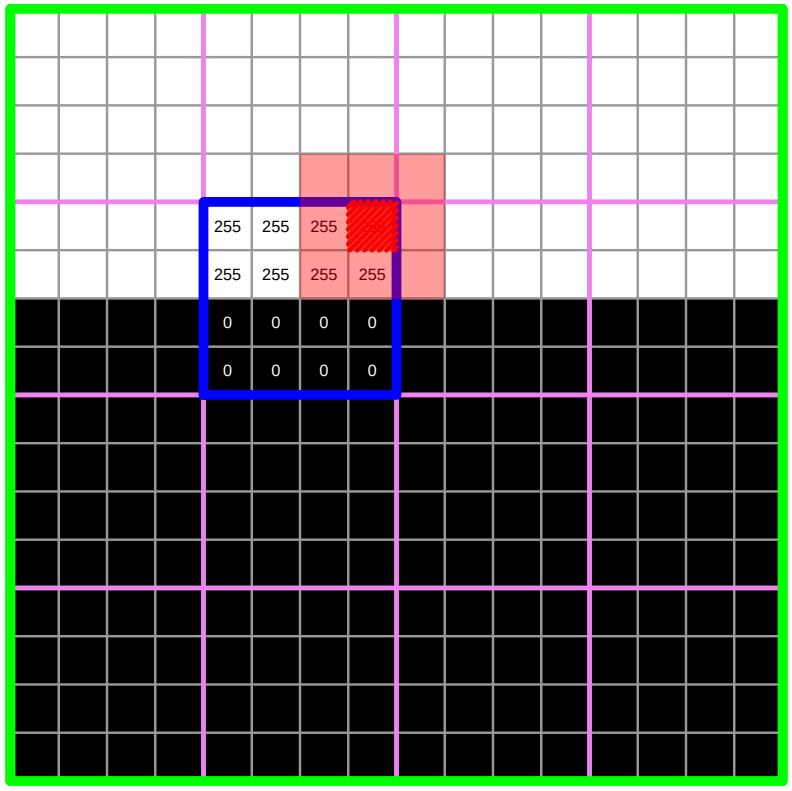
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



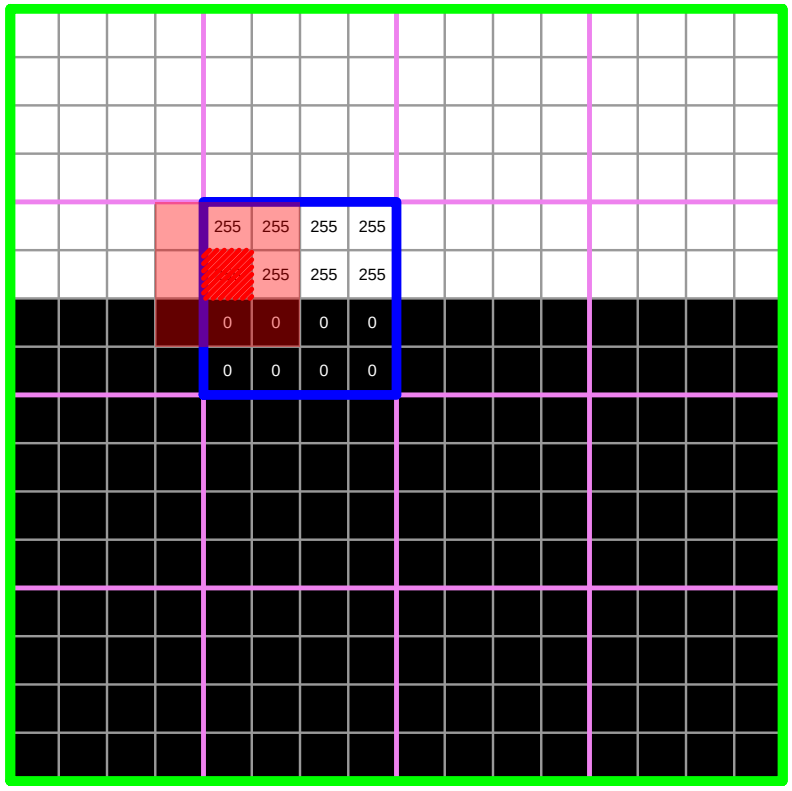
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



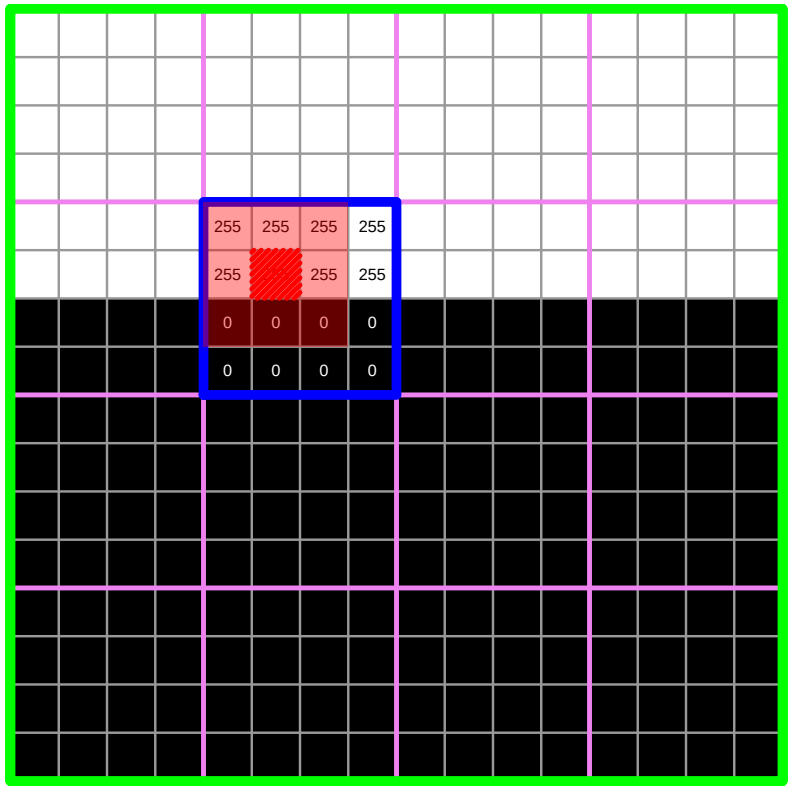
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



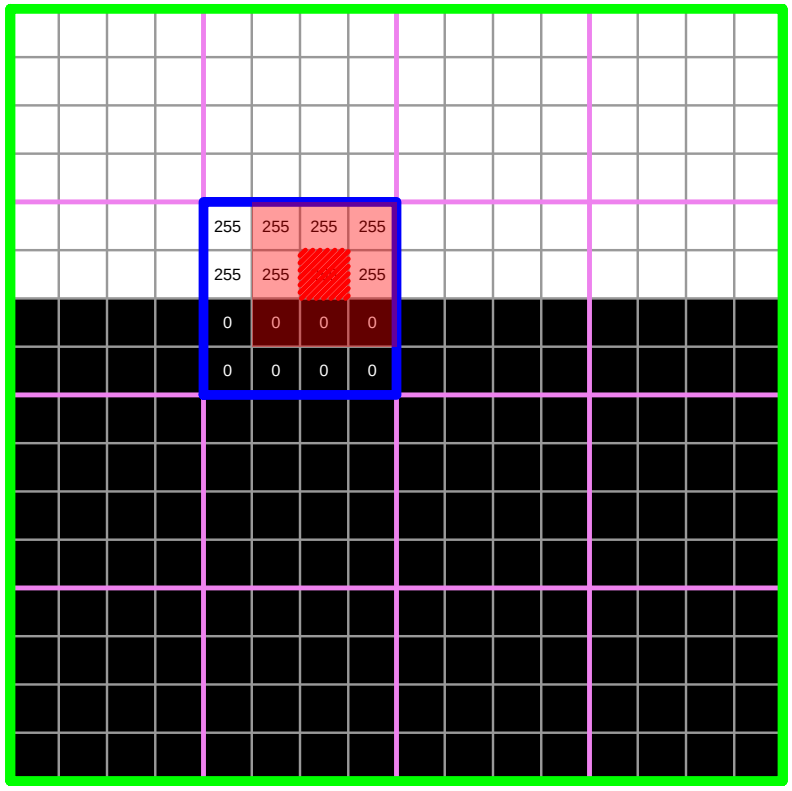
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



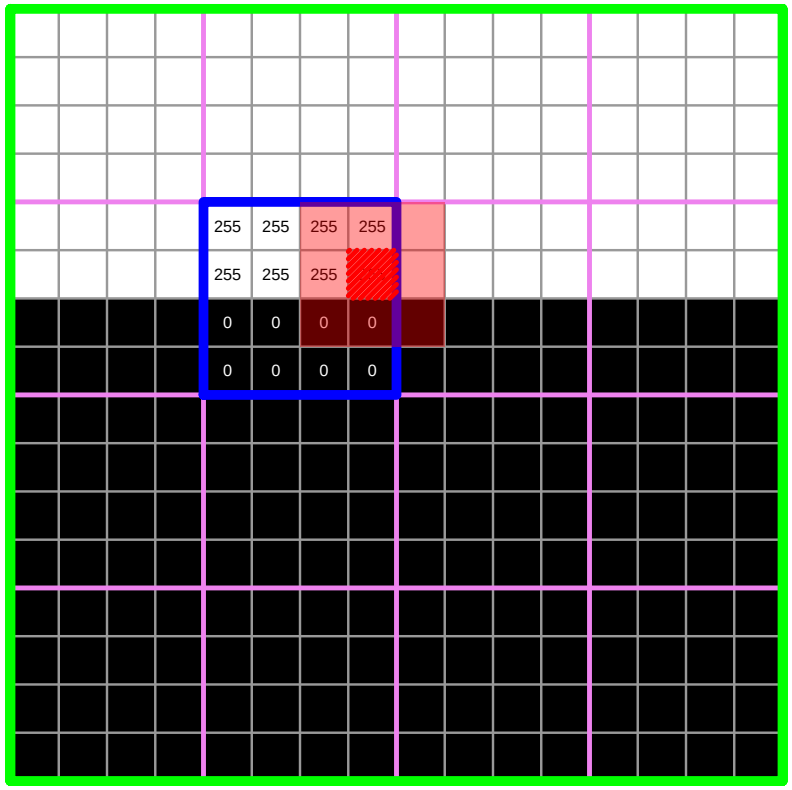
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



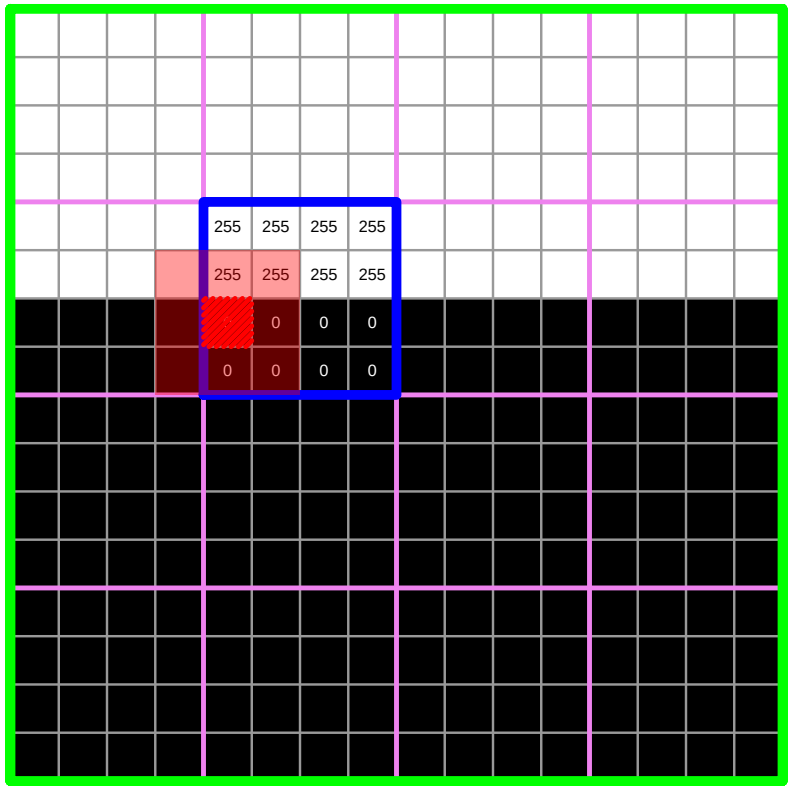
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



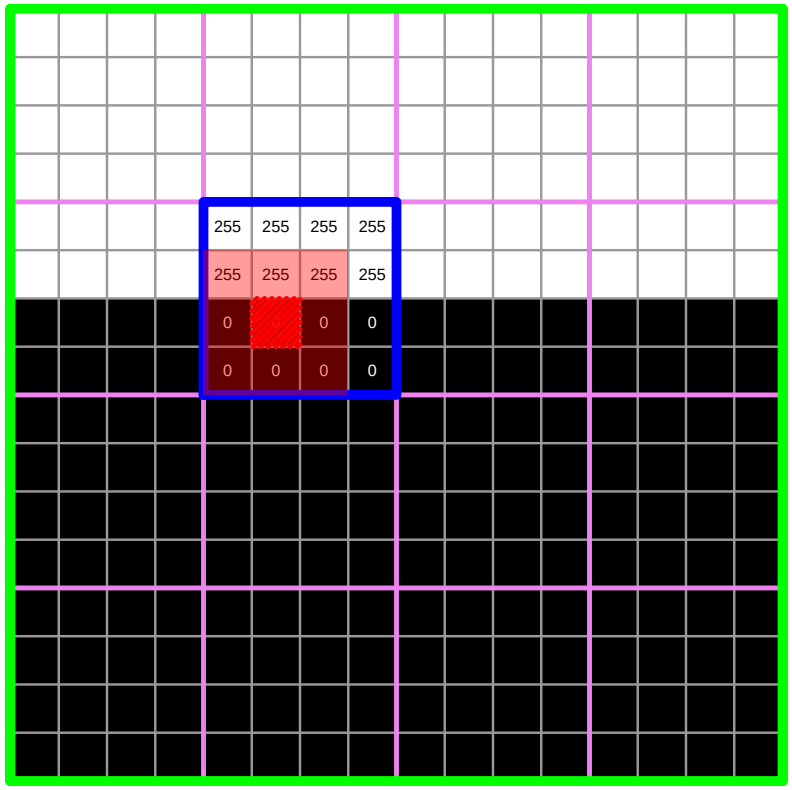
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



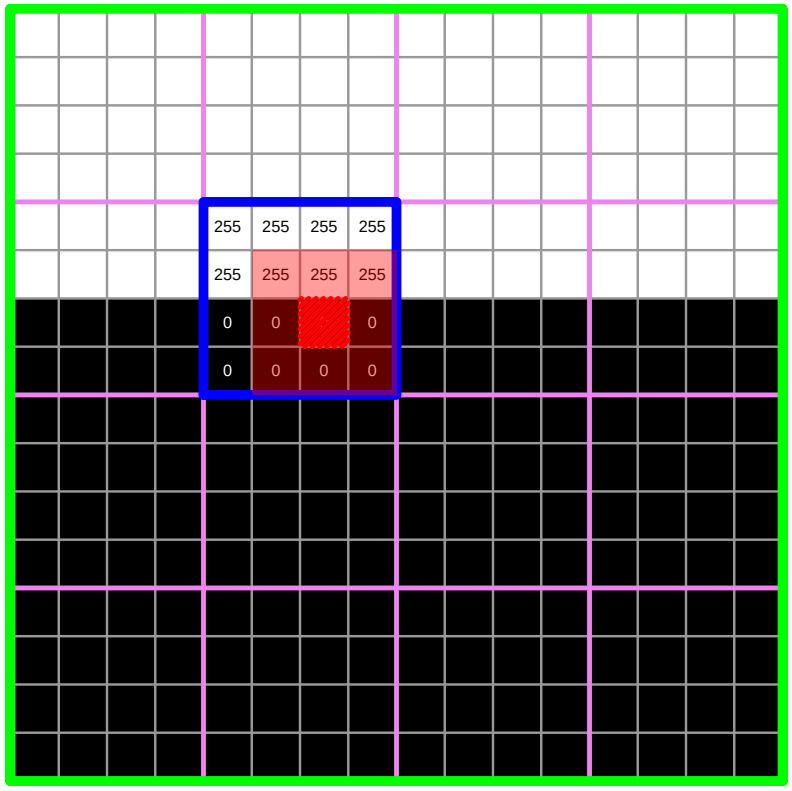
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



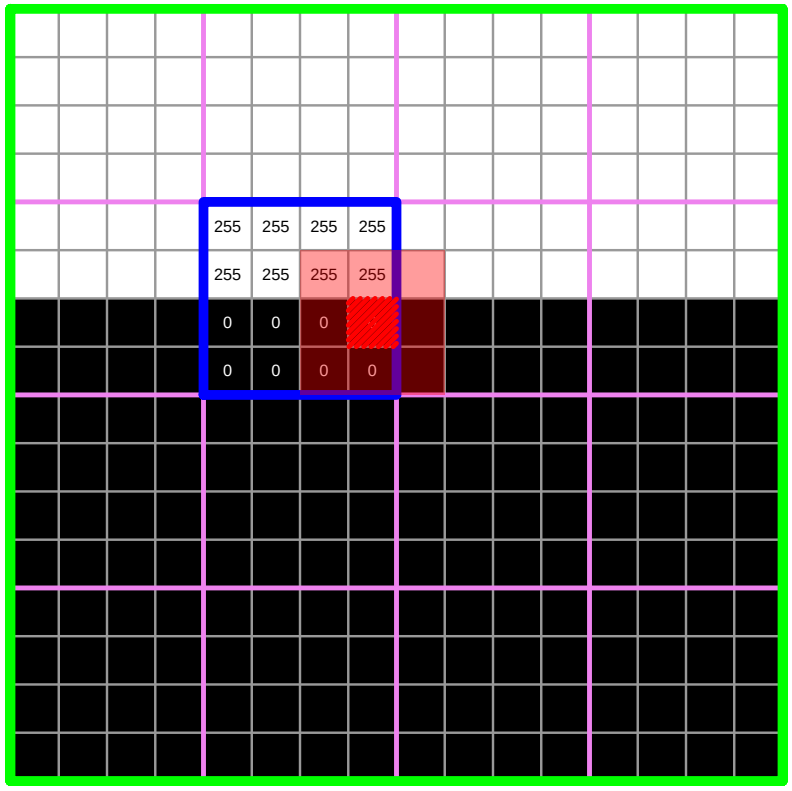
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



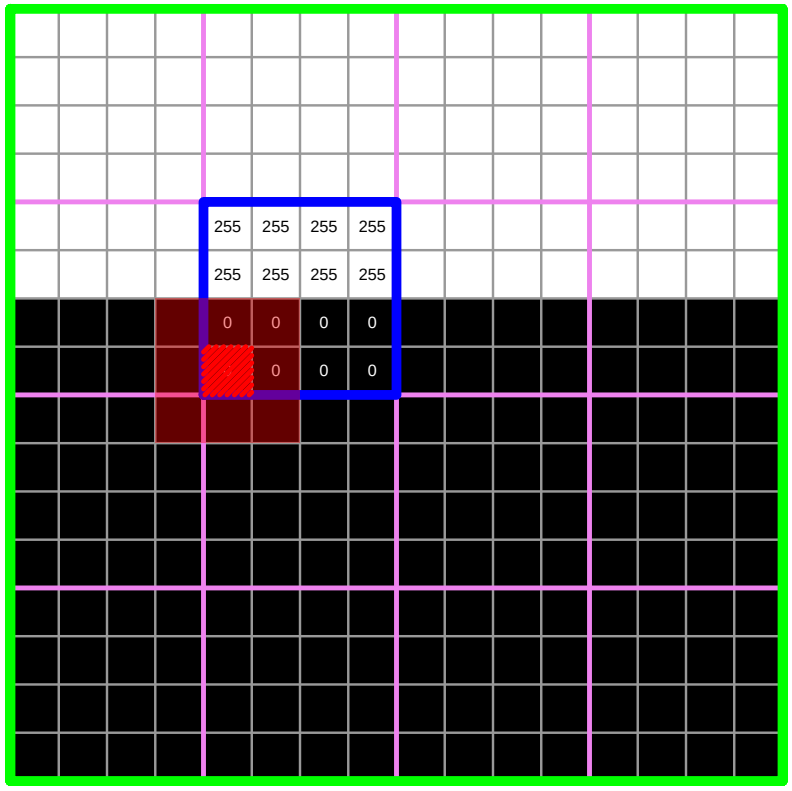
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



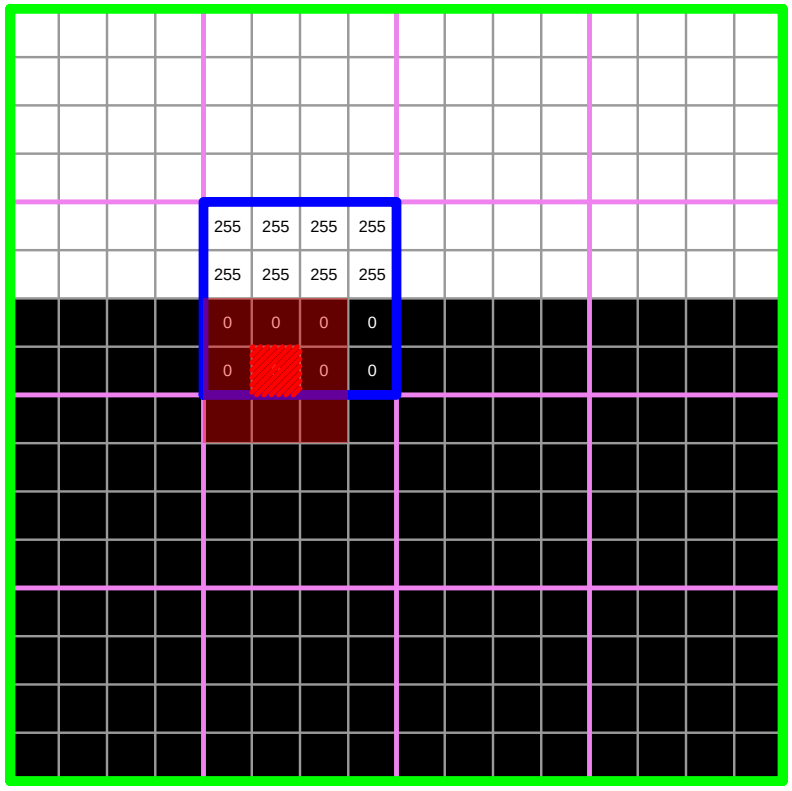
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



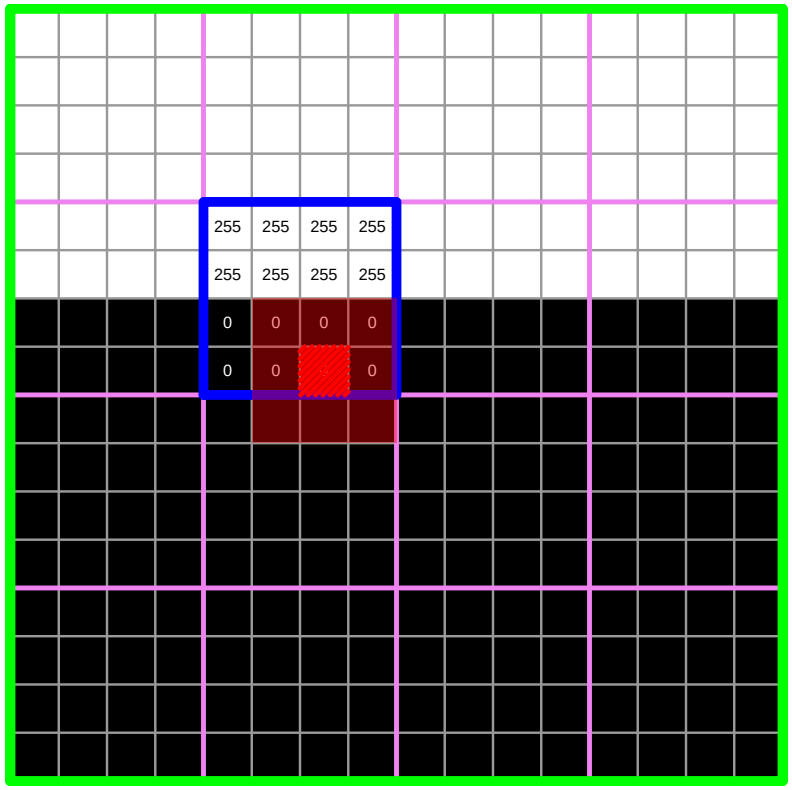
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



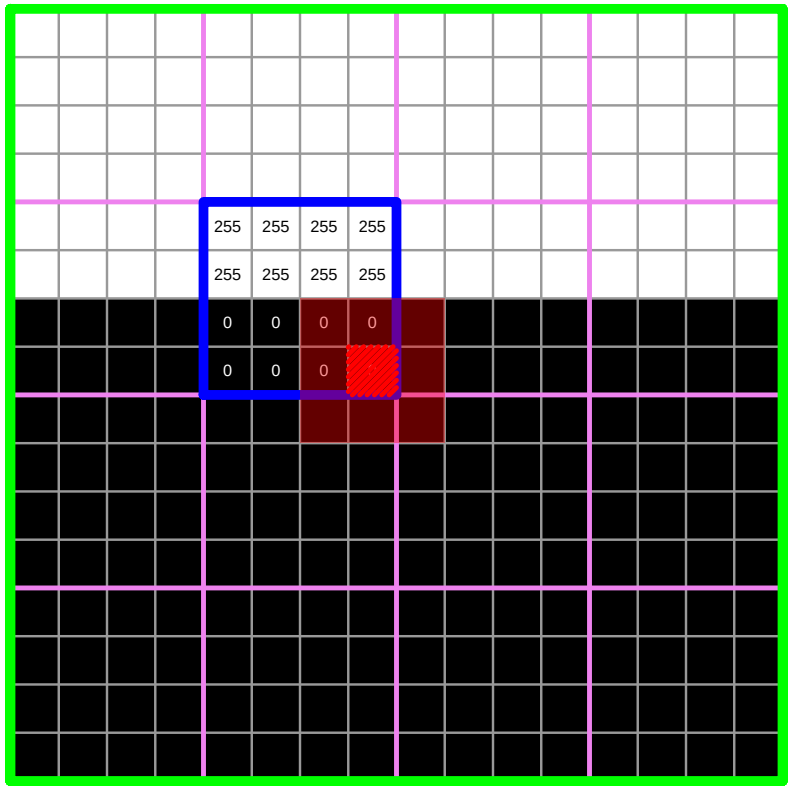
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



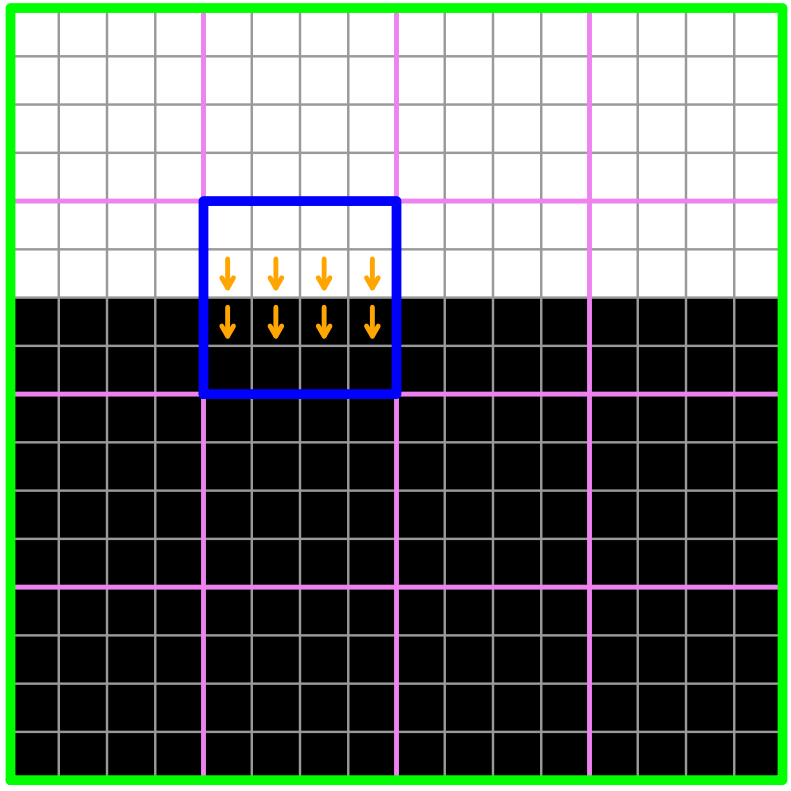
Feature identification: Feature extraction (cont.)

- Slide a kernel along pixels to estimate gradients:



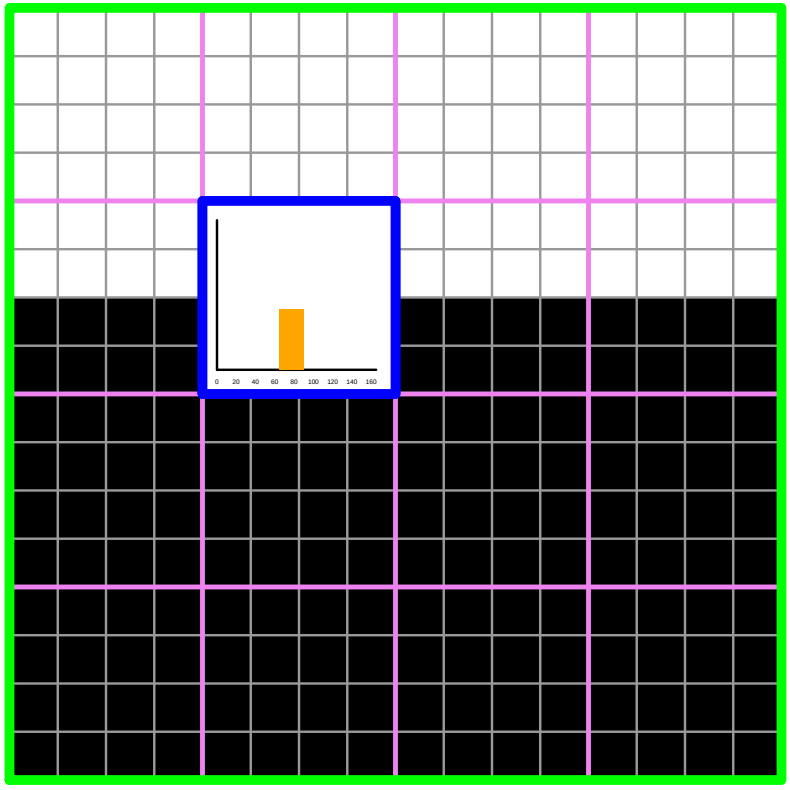
Feature identification: Feature extraction (cont.)

- Summarize gradient orientations and magnitudes

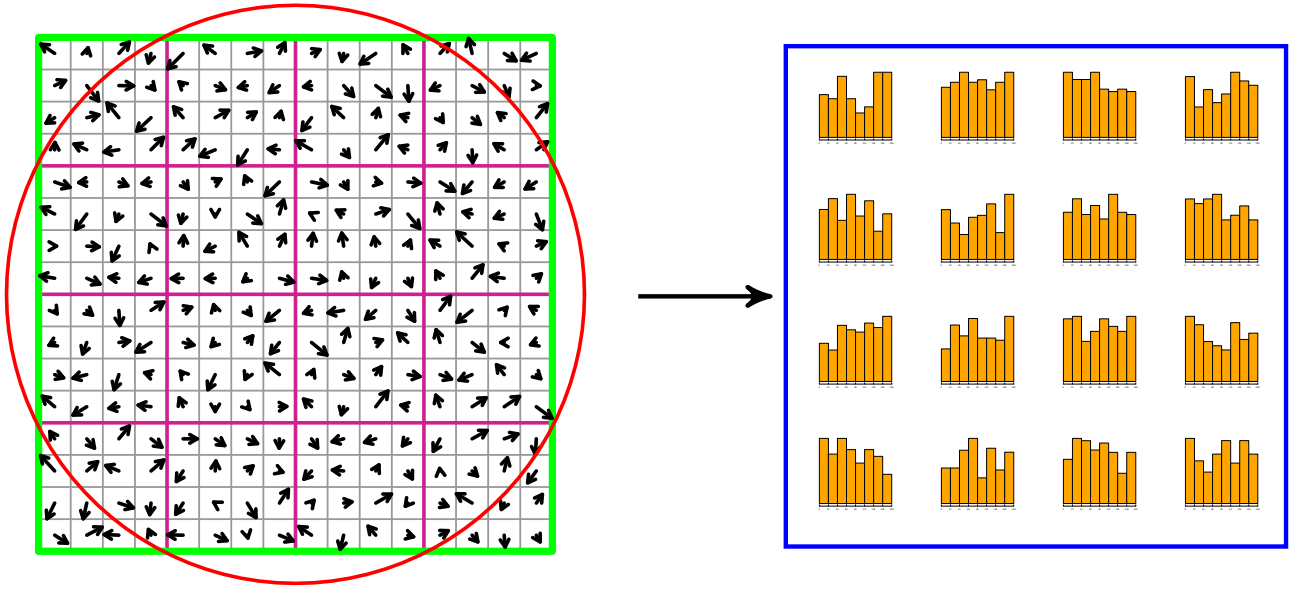


Feature extraction: Step 4

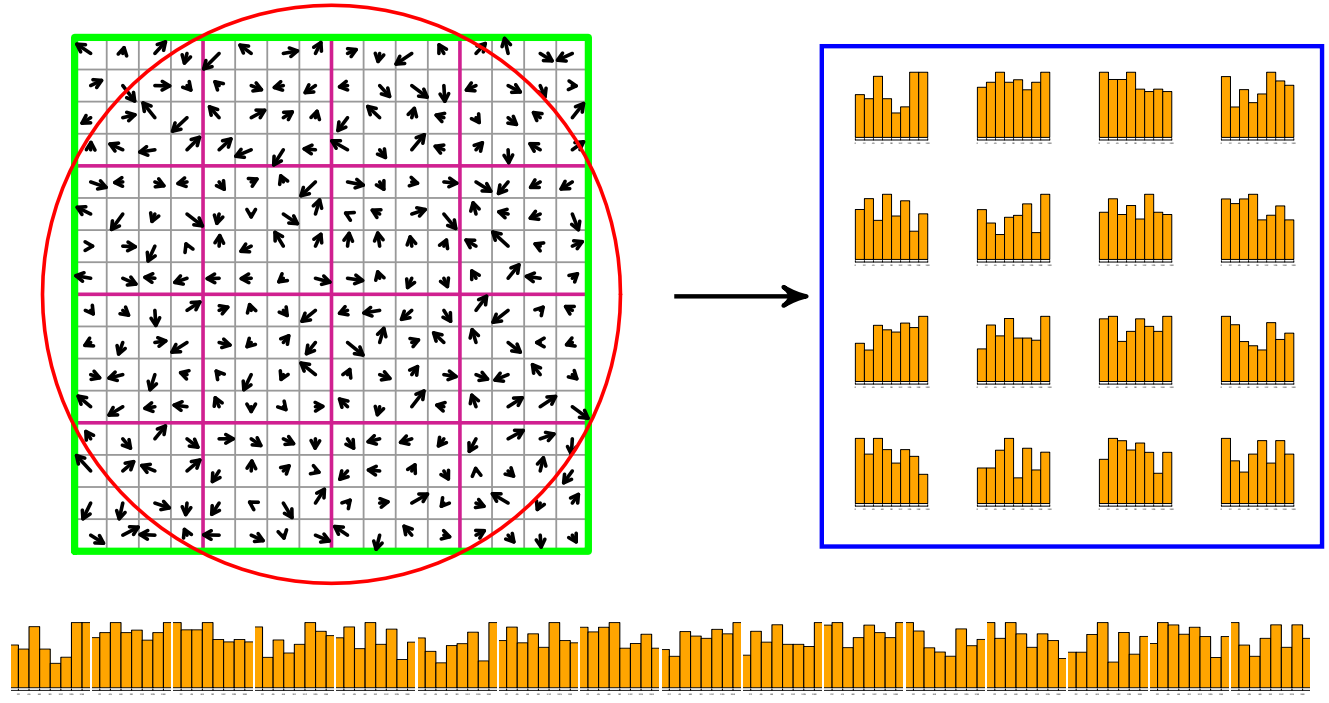
- Build HoG: 8-bins for the multiple orientations, weighted by magnitude and distance to the key point



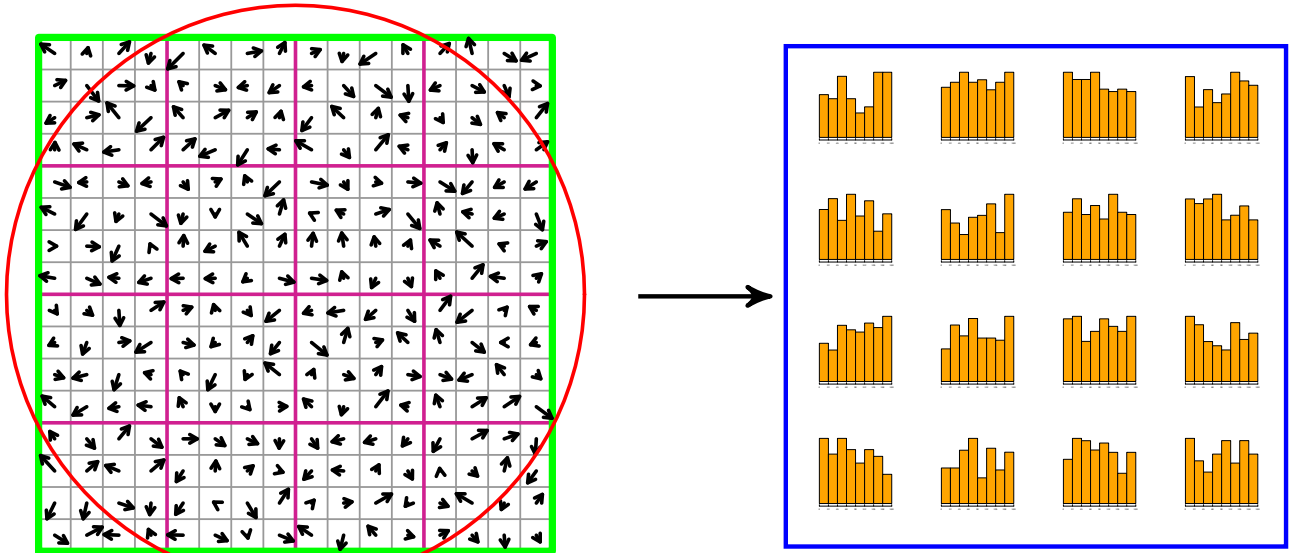
Feature extraction: From pixels to Vectors



Feature extraction: From pixels to Vectors

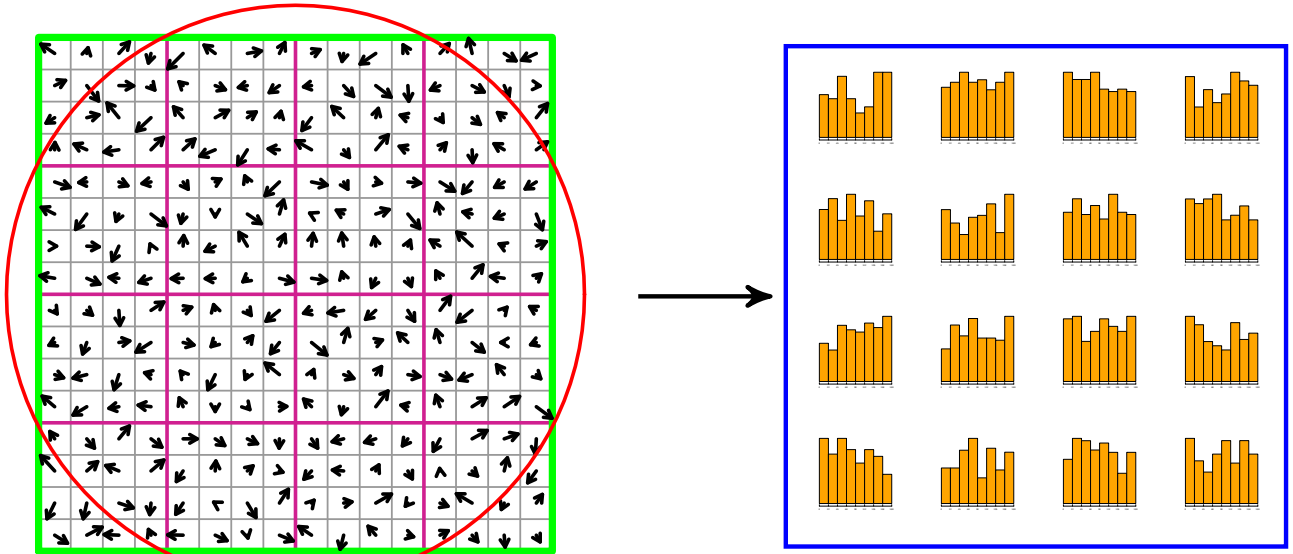


Feature extraction: From pixels to Vectors



$$\begin{bmatrix} 1200 & 2700 & 1230 & \cdots & 4398 & 786 & 3214 \end{bmatrix}$$

Feature extraction: From pixels to Vectors

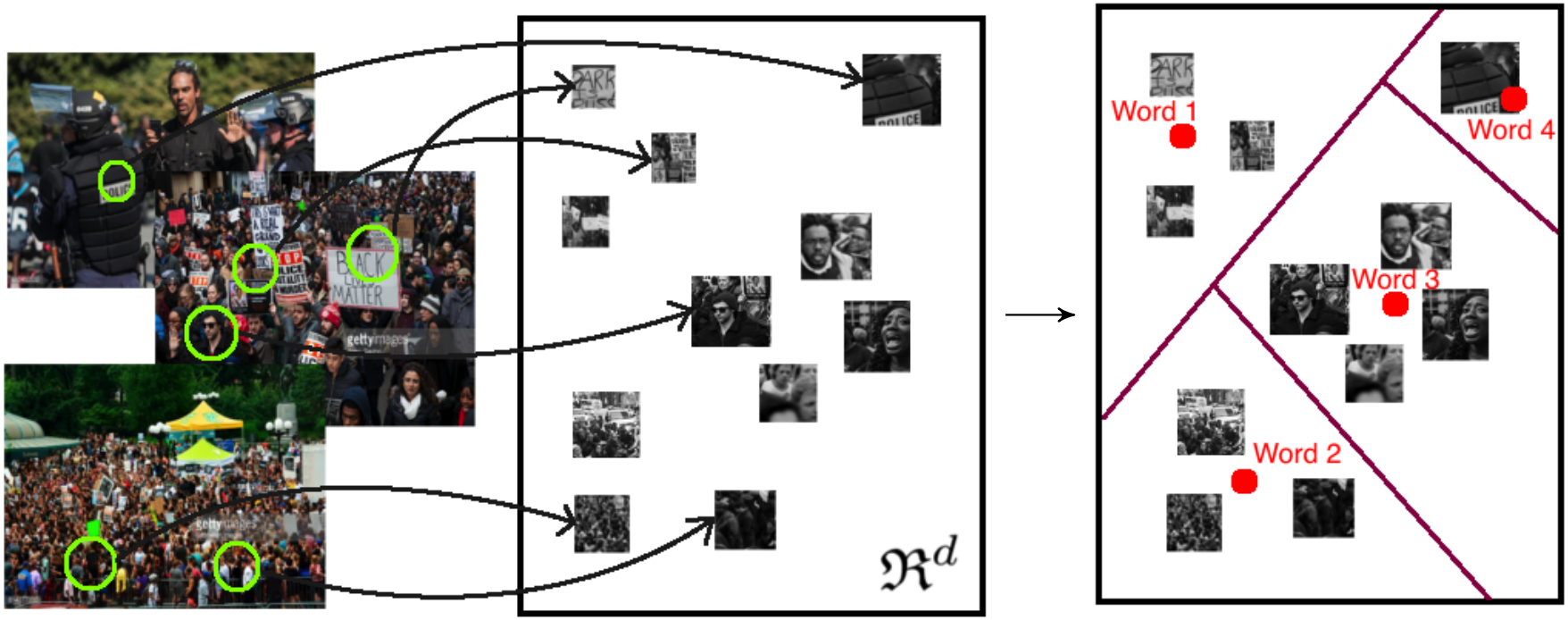


$$\begin{bmatrix} 1200 & 2700 & 1230 & \cdots & 4398 & 786 & 3214 \end{bmatrix}$$

Each point represented with a vector with 128 elements:
8-bins per histogram $\times$ 16 cells

VISUAL VOCABULARY DEFINITION

- Cluster the features and take the centroid
- Mini-batch *k*-means



DEFINING A VISUAL WORD

- Intuitively: Clusters of mini-patches
- Formally: the vector representing the centroid of the cluster

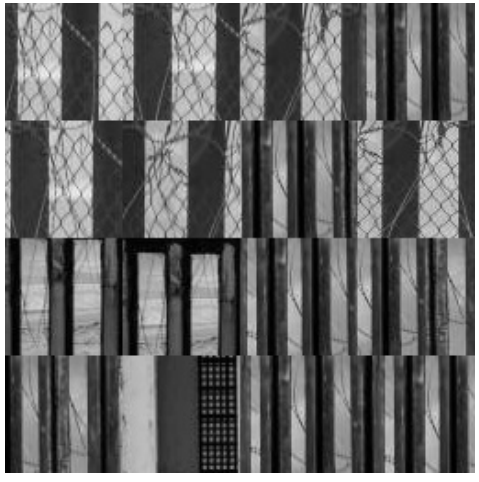


Image-Visual Word Matrix

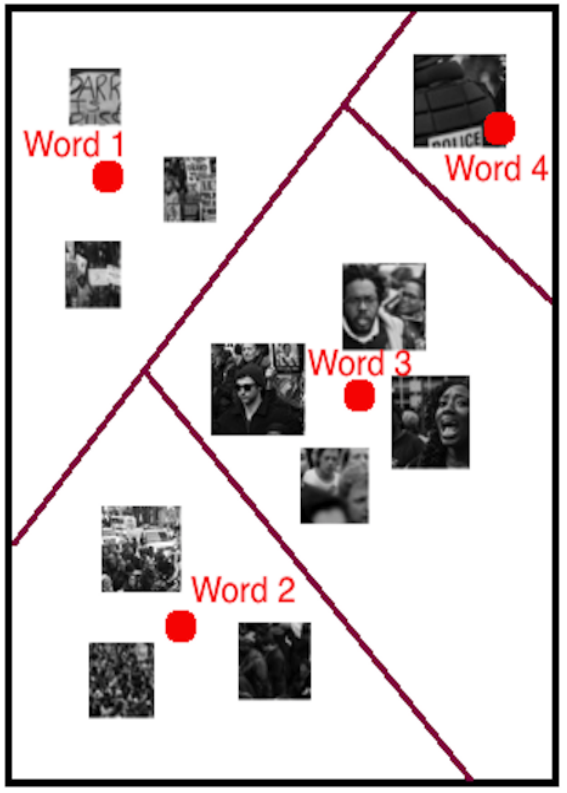


Image-Visual Word Matrix

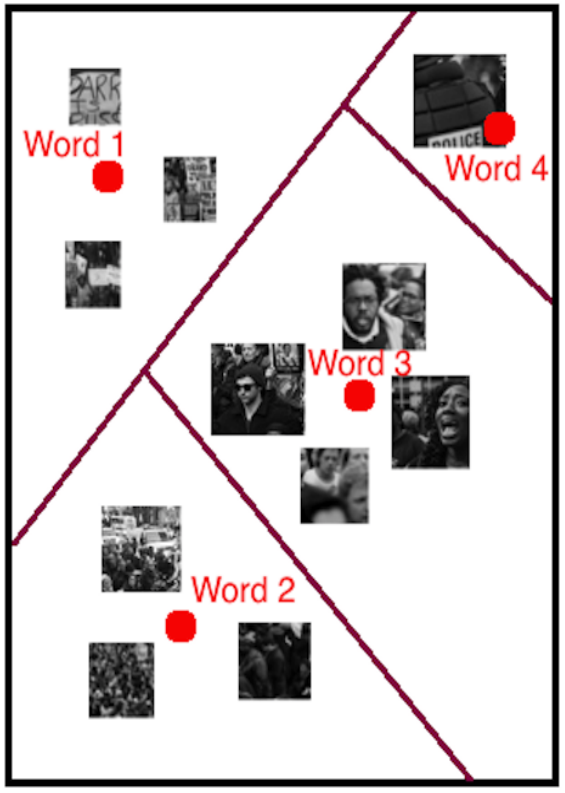


IMAGE-VISUAL WORD MATRIX

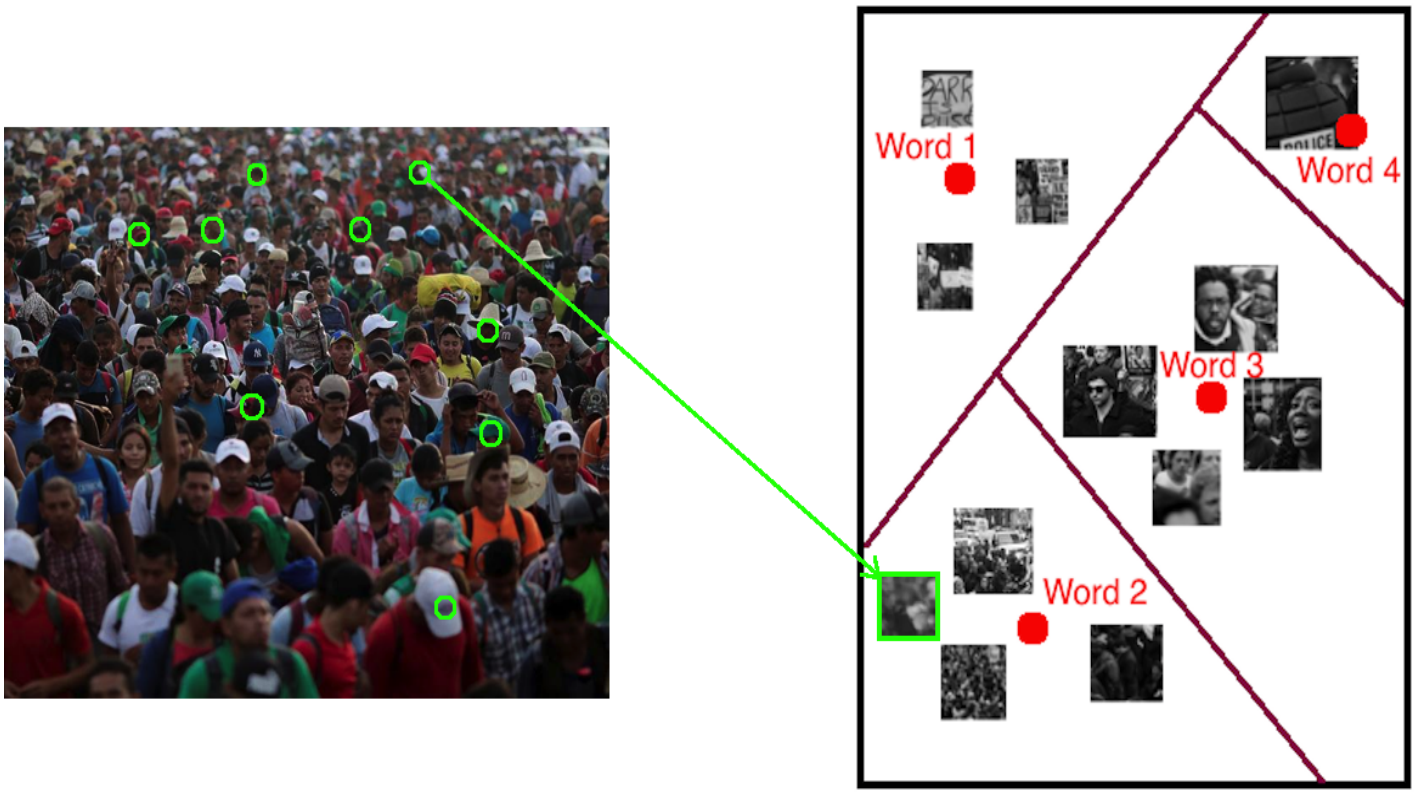


Image-Visual Word Matrix

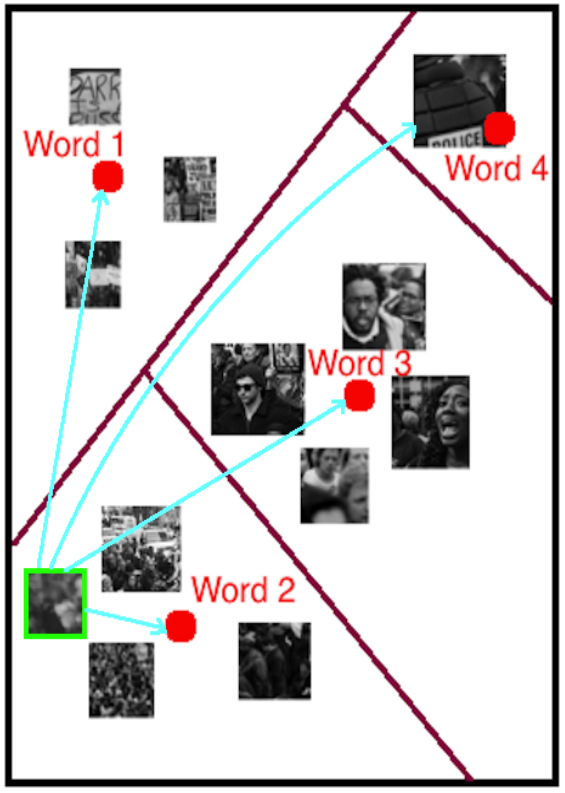


IMAGE-VISUAL WORD MATRIX

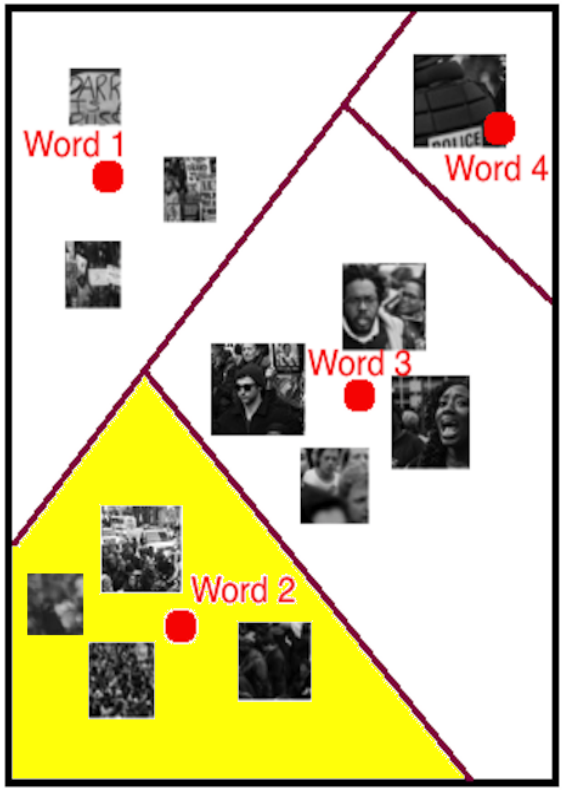


Image-Visual Word Matrix: Some considerations

- Clusters = visual words
- Euclidean distance to determine similarity
- Every key-point is assigned to one word (something to consider)

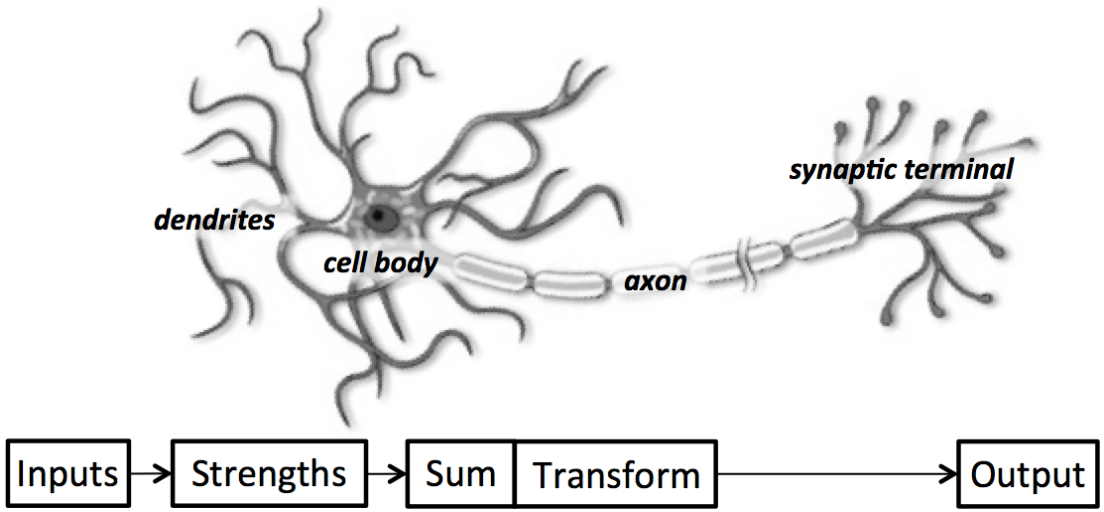
Convolutional Neural Networks

SEEING LIKE A HUMAN

- Modern computer vision systems are meant to emulate how human brains transform sensual stimuli into conceptual understanding
- The process allows computers to set their own set of rules to classify information

SEEING LIKE A HUMAN

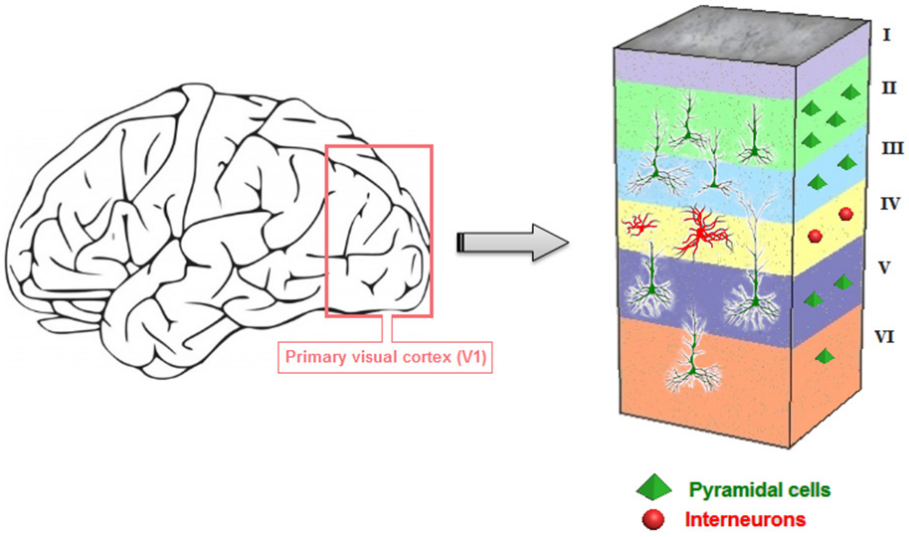
- Modern computer vision systems are meant to emulate how human brains transform sensual stimuli into conceptual understanding
- The process allows computers to set their own set of rules to classify information



Credit: *Buduma (2017)*

- Neurons are the core unit of brains.
- They receive information from other neurons, process the input, and send the result to other cells for further processing.

SEEING LIKE A HUMAN, CONT.



Credit: *Bachatene, Bharmauria and Molotchnikoff (2012).*

- Neurons are organized into layers.
- Every layer breaks down the signal into small pieces, allowing each of its neurons to focus on a unique piece of information.
- The first layers identify basic visual patterns, intermediate layers transform patterns into shapes, and the last layers convert shapes into objects.

Convolutional Neural Networks (CNNs)

- Convolutional Neural Networks (CNNs) is a supervised learning algorithm to classify images
- CNNs gradually learn what visual features of the image are more important in a classification task by transforming the image into multiple representations or *feature maps*.
- CNNs are organized into multiple layers. Each layer contains multiple representations of the original image through maps of visual features such as edges, blobs or color combinations.
- The part of learning and reaching a semantic concept that humans conduct by trial and error is achieved through the training, validation and testing procedures in CNNs.

NETWORK STRUCTURE

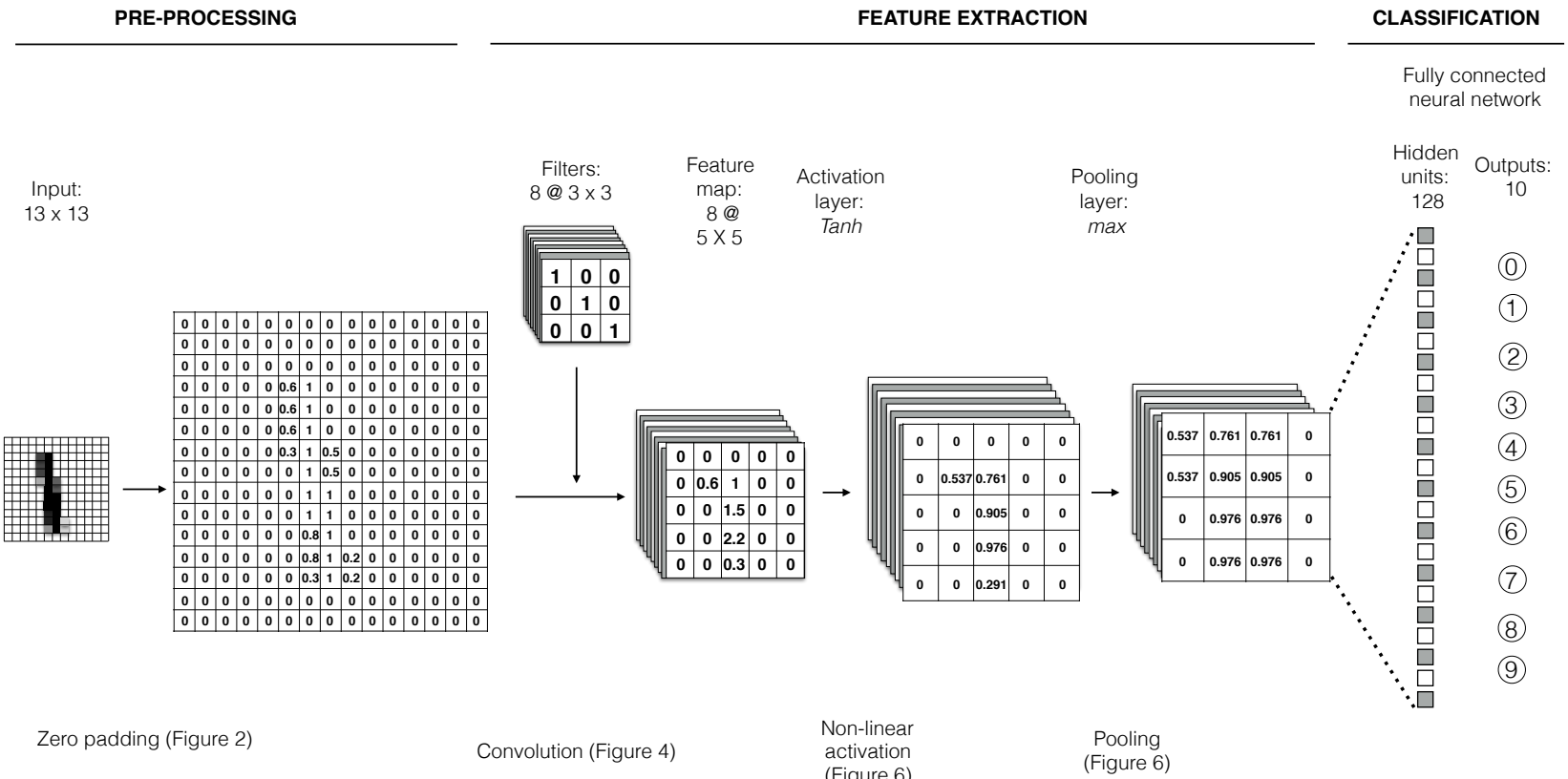
- GOAL: learn the features associated w/ outcomes

NETWORK STRUCTURE

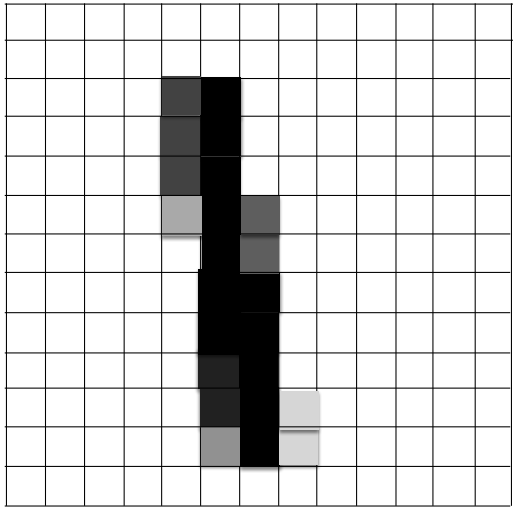
- GOAL: learn the features associated w/ outcomes
- Translation: obtain “coefficients” [weights in feature maps]
- Mainly, a data reduction technique → **Why?**

NETWORK STRUCTURE

- GOAL: learn the features associated w/ outcomes
- Translation: obtain “coefficients” [weights in feature maps]
- Mainly, a data reduction technique → **Why?**
- Not a black-box! → Optimization of error



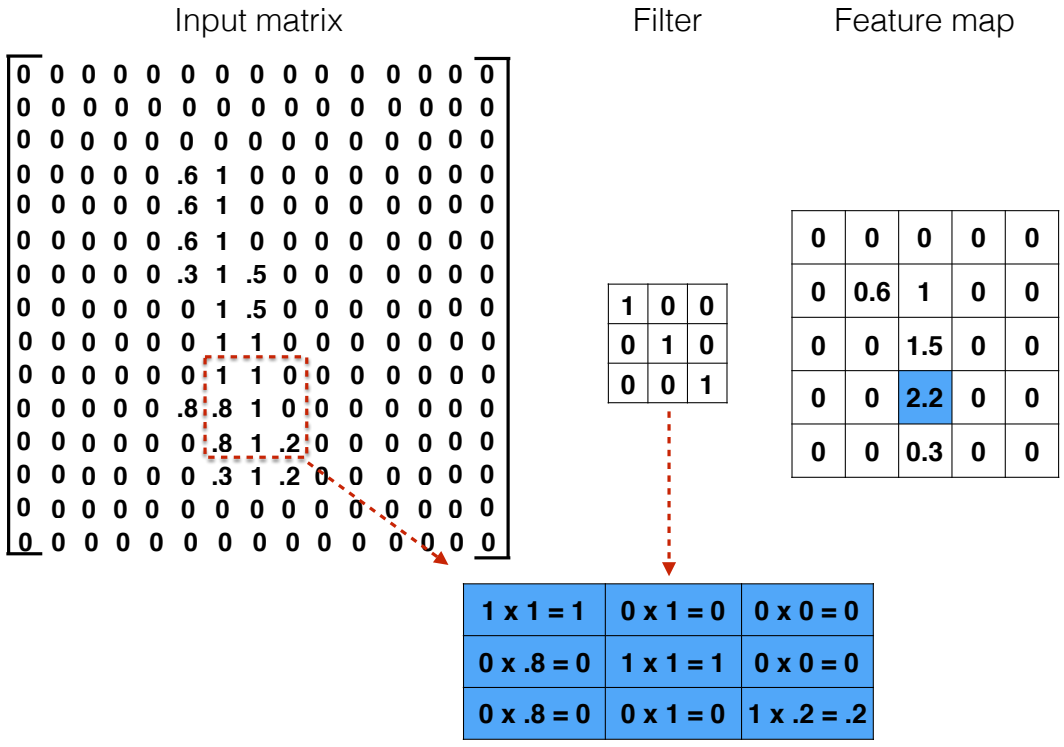
REPRESENTING IMAGES



$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & .6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & .6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & .6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & .3 & 1 & .5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & .5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & .8 & .8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & .8 & 1 & .2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & .3 & 1 & .2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

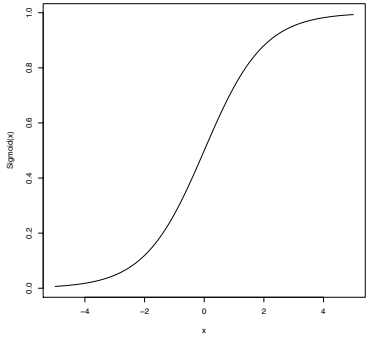
The image is transformed into a numerical matrix, where each element represents the value of a specific pixel of the image measured as light intensity (in grayscale images) or color intensity (in color images).

FEATURE EXTRACTION

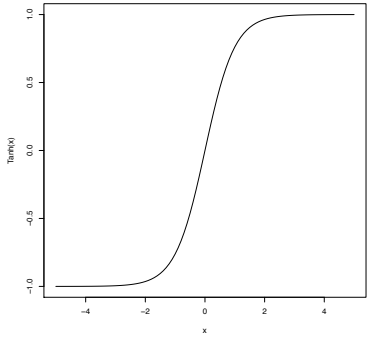


Filters are matrixes made of *weights*, that maximize or minimize the "intensity" of a pixel. Every filter slides through each 3 x 3 pixel area of the image, and computes the dot product of the region. The result is recorded on a smaller matrix to create *feature maps*. Intuitively, we want to detect whether and where a feature represented by a filter is prominent in the image.

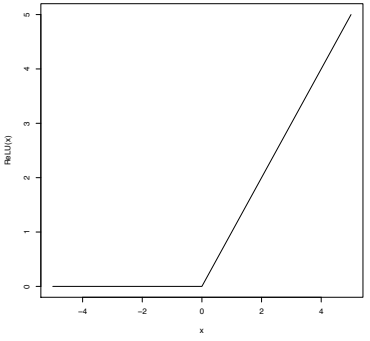
ACTIVATION FUNCTIONS



(a) $Sigmoid(x) = \frac{1}{1+e^{-x}}$



(b) $Tanh(x) = \frac{2}{1+e^{-2x}}$



(c) $ReLU(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{otherwise.} \end{cases}$

We add non-linearity by including an *activation layer*.

POOLING STAGE

Non-linear activation: Tanh(x)

max pooling

0	0	0	0	0
0	0.6	1	0	0
0	0	1.5	0	0
0	0	2.2	0	0
0	0	0.3	0	0

0	0	0	0	0
0	0.537	0.761	0	0
0	0	0.905	0	0
0	0	0.976	0	0
0	0	0.291	0	0

0.537	0.761	0.761	0
0.537	0.905	0.905	0
0	0.976	0.976	0
0	0.976	0.976	0

Once the activation map shows non-linear outputs, we reduce its dimensionality using a *pooling layer*. A pooling layer shrinks the size of the matrix while keeping the most important information in the feature map.

LEARNING

- The last stage of the network involves the classification of the image. The way in which the CNN learns the features that correlate to each outcome follows a procedure called back-propagation.

More on back-propagation

EXAMPLE: HANDWRITING RECOGNITION

8 RESULTADOS DE LA VOTACIÓN DE DIPUTADOS FEDERALES *(Escriba los votos para cada partido político, candidatos no registrados y votos nulos, súmelos y escriba el resultado en TOTAL). En caso de no recibir votos para algún partido o candidato escriba ceros.*

PARTIDO POLÍTICO	RESULTADOS DE LA VOTACIÓN DE DIPUTADOS FEDERALES DE MAYORÍA RELATIVA (Con letra)	(Con número)
PAN		038
PRI		072
PRD		003
VERDE		014
PT		003
MOVIMIENTO CIUDADANO		002
alianza		236
morena		002
NUEVA ALIANZA		020
encuentro social		001
CANDIDATOS NO REGISTRADOS		000
VOTOS NULOS		319
TOTAL		710

9 ¿ES IGUAL LA CANTIDAD DEL APARTADO 6 CON EL TOTAL DE LOS VOTOS DEL APARTADO 8 ? (SÍ) (NO) *(Marque con "X")*

10 ¿SE PRESENTARON INCIDENTES DURANTE EL ESCRUTINIO Y CÓMPUTO DE LA ELECCIÓN DE DIPUTADOS FEDERALES? (SÍ) (NO) *(Marque con "X")*

DESCRIBA BREVEMENTE: _____

_____ EN SU CASO, SE ESCRIBIERON EN _____ HOJA(S) DE INCIDENTES, MISMA(S) QUE SE *(Con número)*

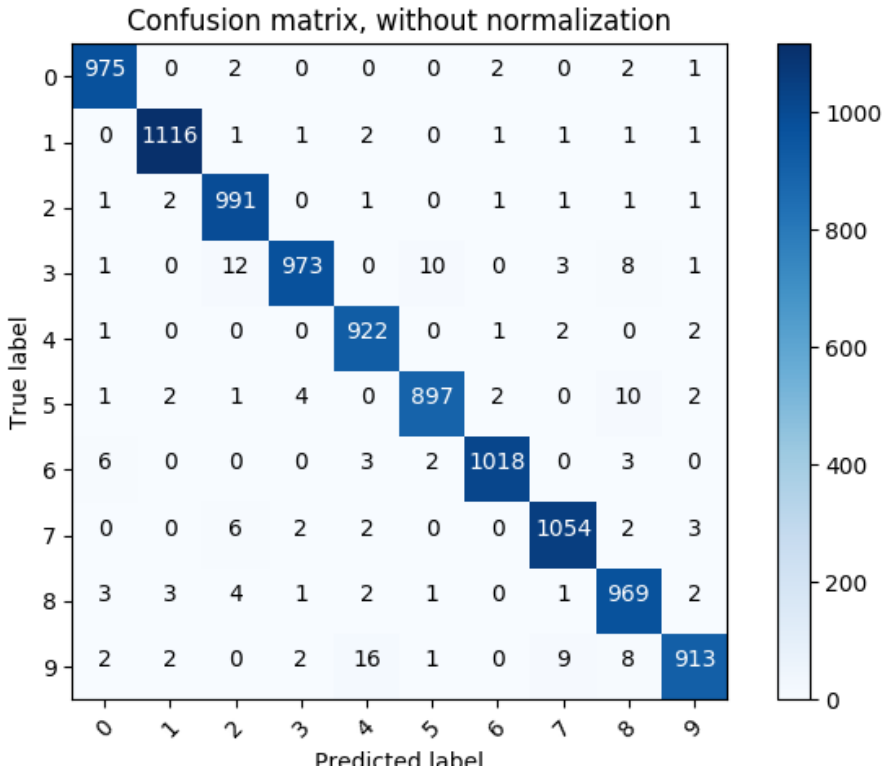
ANEXA(N) A LA PRESENTE ACTA.

Train and test

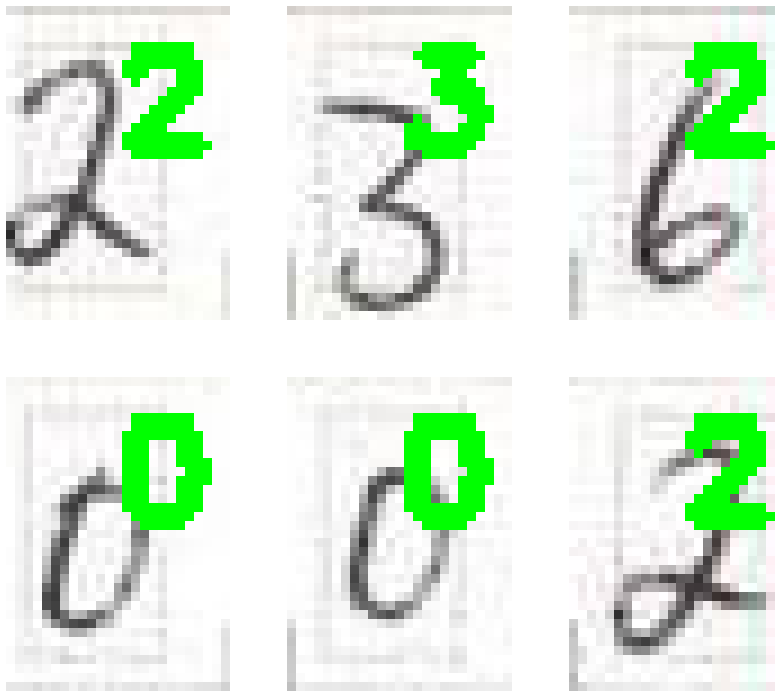
- MNIST dataset: 70,000 samples of numbers
- Train: 60,000
- Test: 10,000
- Accuracy rate: Only MNIST=98.3%, Real data= 96.4%

TRAIN AND TEST

- MNIST dataset: 70,000 samples of numbers
- Train: 60,000
- Test: 10,000
- Accuracy rate: Only MNIST=98.3%, Real data= 96.4%



PREDICT THE NUMBERS



CHALLENGES AND RECOMMENDATIONS

- Prevent overfitting
  - Increase number of training images
  - Data augmentation
  - Dropout random neurons
- Optimize your training set
  - Active learning: Informativeness vs. Representativeness
  - Class balance
  - “Denoise” images
  - Batch normalization
  - CAUTION: Bias training
- Post-CNN diagnosis
  - Know your training, testing and out-of-sample data
  - Always check mislabeled examples: validate, validate, validate...
  - Diagnosis

IN PRACTICE

- Design your structure and create your training data
  - AWS machines, HPC or GPUs [computational power needed]
  - Pre-trained architectures in Google, Amazon, etc.
  - Creating training data: `imglab`
- Pre-canned image detection with API access
  - GoogleVision: `https://cloud.google.com/vision/`, Amazon, Microsoft
    - Labels found in each picture
    - Face detection
    - Emotions
    - Sensitive content

OBJECT DETECTION: COVERS OF NEWSPAPERS

Full set of images

Only Women's March images

FACE DETECTION AND EMOTIONAL CONTENT

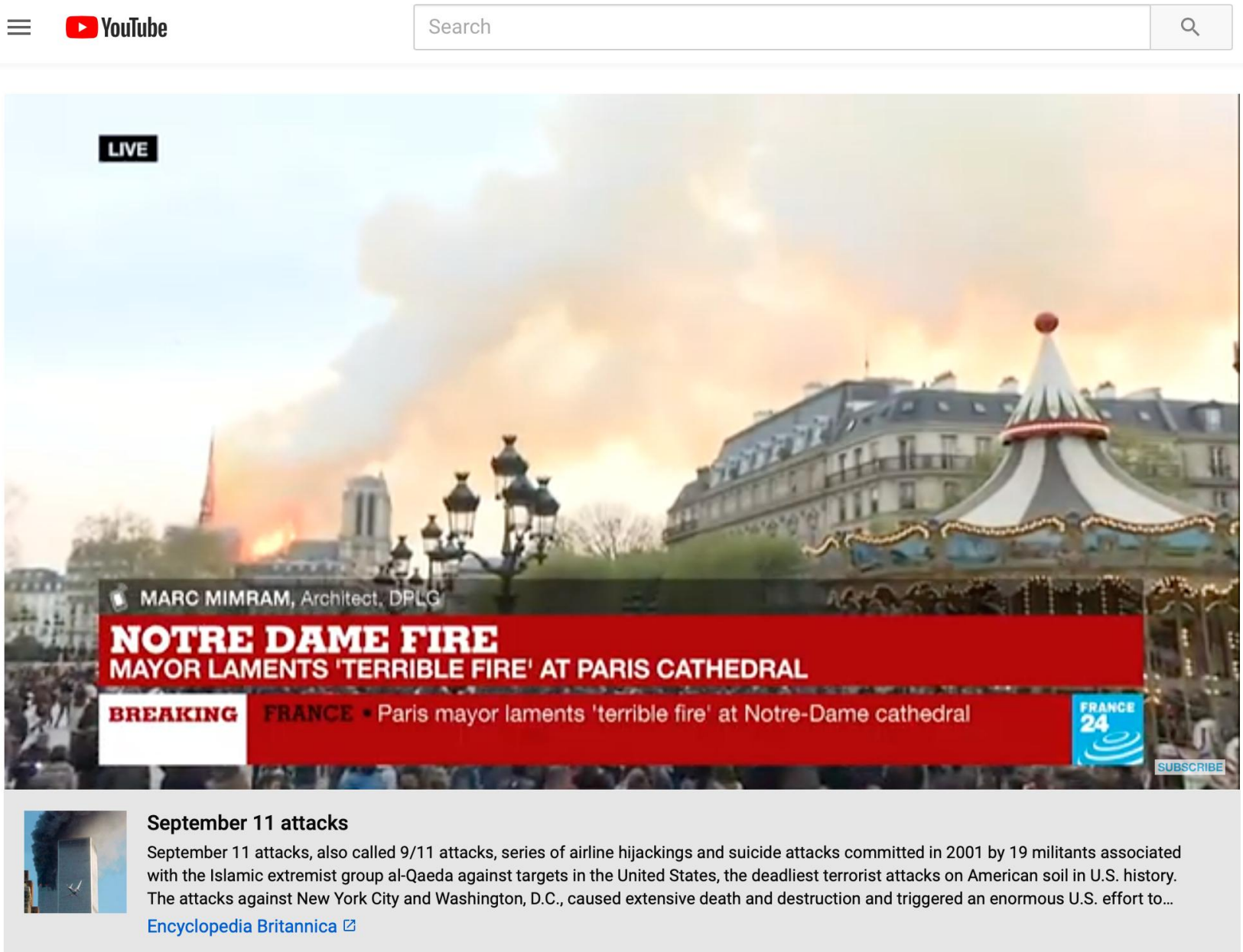
Good results with little effort...

Face detection and emotional content

Good results with little effort...

...but also tons of errors

UNINTENDED BUT DANGEROUS CONSEQUENCES

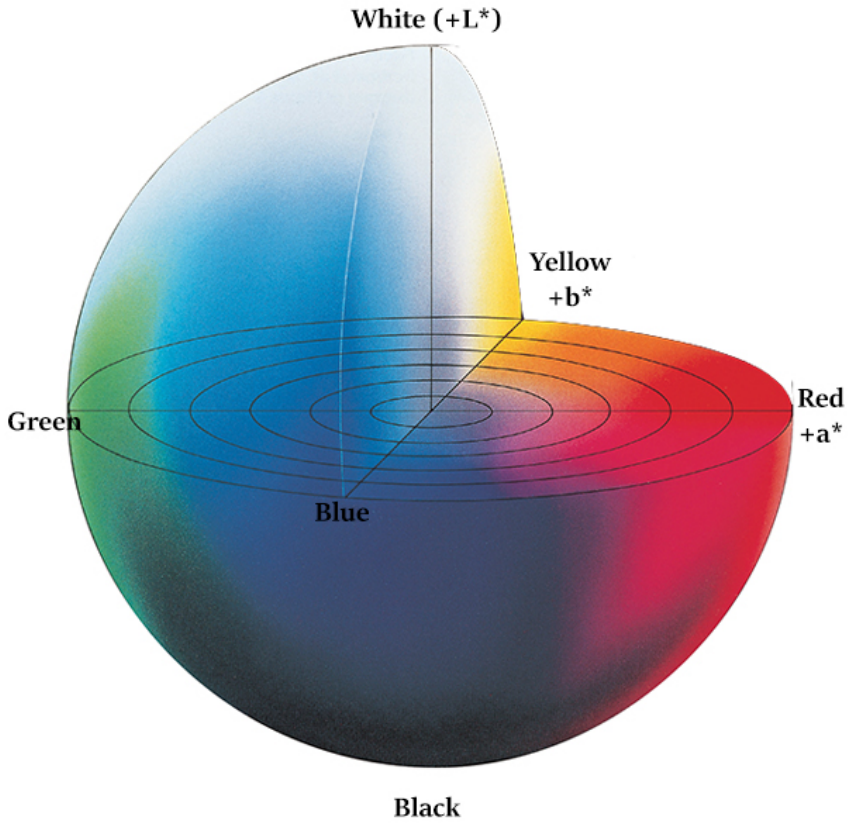


Other considerations

- Pay attention to your training data
- Consider using transfer learning or pre-trained architectures
- Take advantages of APIs
- Understand the limitations and scope of the method

L*a*b COLOR SPACE

- L* = lightness
- a* = chromaticity coordinate (red axis)
- b* = chromaticity coordinate (blue axis)



Backpropagation

- Neuron j's classification outcome: y_j.
- Target outcome: t_j.
- Sum of prediction errors $E = \frac{1}{2} \sum_{j \in 10} (t_j - y_j)^2$
- Error derivative function of last layer: $\frac{\partial E}{\partial y_j} = -(t_j - y_j)$

Similarly, we can express the error derivatives in terms of the logit of the neuron, z_j:

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j} = y_j(1 - y_j) \frac{\partial E}{\partial y_j}$$

BACKPROPAGATION, CONT.

- Estimate how the neuron outcomes in layer *i* affect the outputs of layer *j* given the weighted connection between both layers, w_{ij}

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial y_i} = \sum_j w_{ij}\frac{\partial E}{\partial z_j} = \sum_j w_{ij} y_j (1 - y_j)\frac{\partial E}{\partial y_j}$$

- These partial derivatives allow us to estimate the contribution of a specific weight to the error term:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}}\frac{\partial E}{\partial z_j} = y_j (1 - y_j)\frac{\partial E}{\partial y_i}$$

$$-\Delta w_{ij} = -\sum_{k \in K} y_i^{(k)} y_j^{(k)} (1 - y_j^{(k)})\frac{\partial E^{(k)}}{\partial y_i^{(k)}}$$