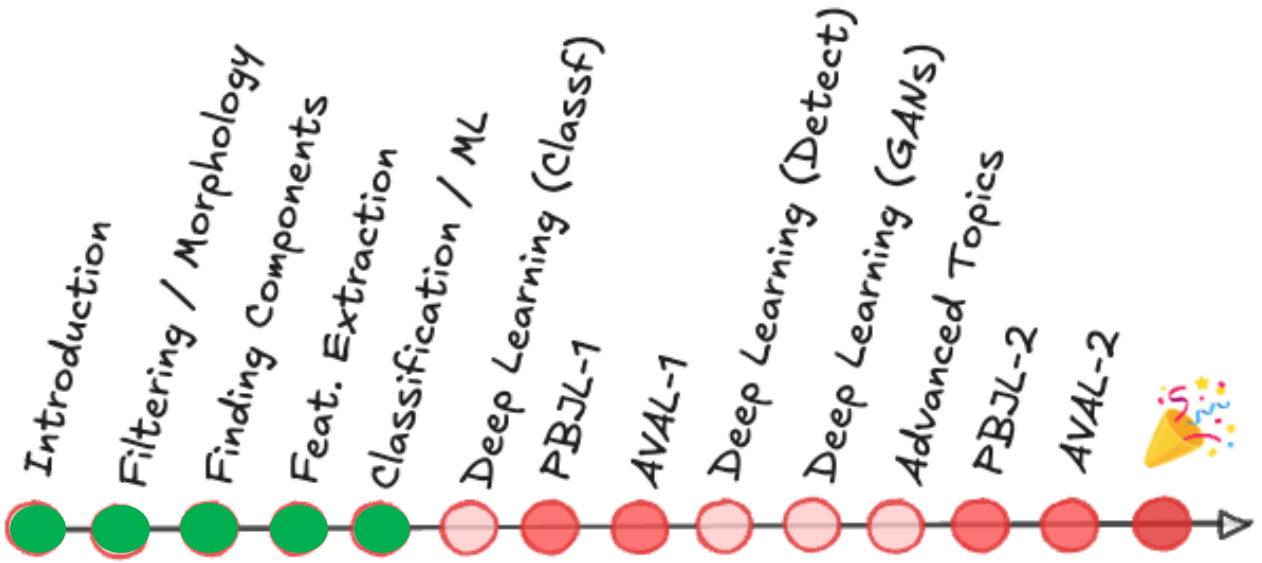
Lecture 05 - Feature Extraction

Prof. André Gustavo Hochuli

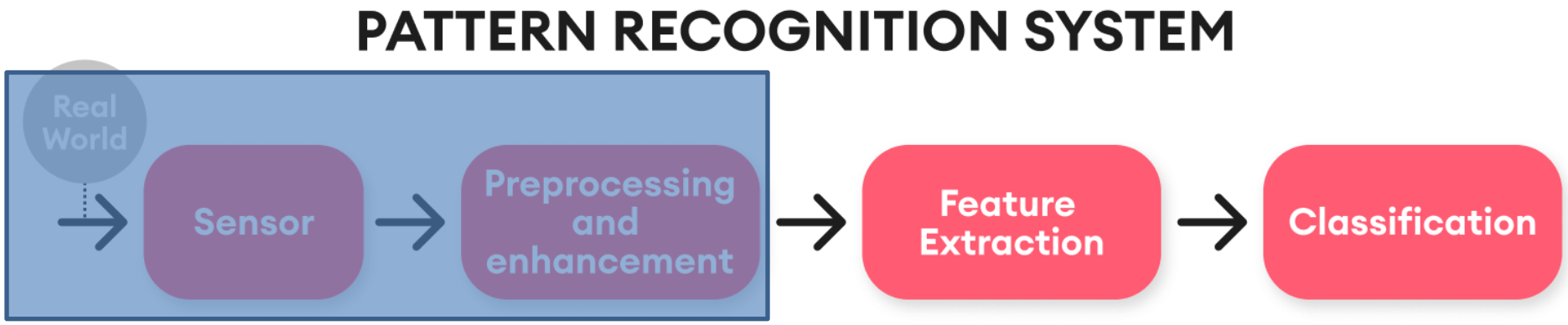
gustavo.hochuli@pucpr.br
aghochuli@ppgia.pucpr.br

Topics

- [Recap] Lecture 04 – Finding Components
- Lecture 05 - Feature Extraction
  - Feature Vector / Embeddings / Representations
  - Feature Space
- Feature Engineering
- Descriptors
- Practice

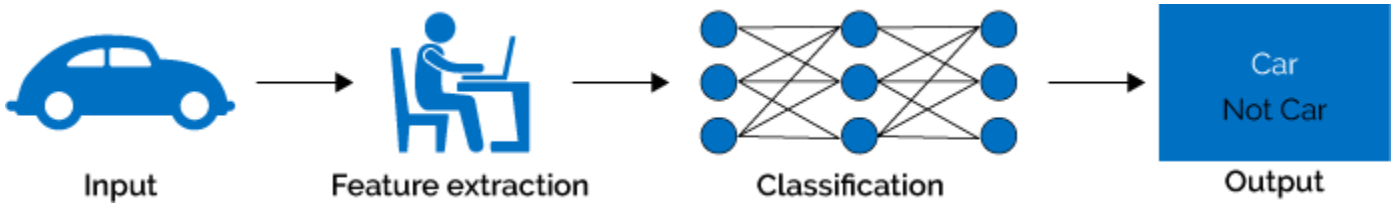


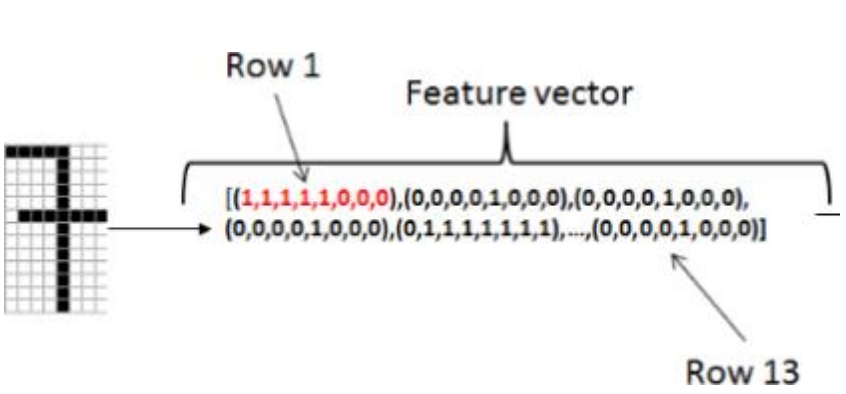
Computer Vision & Pattern Recognition Pipeline

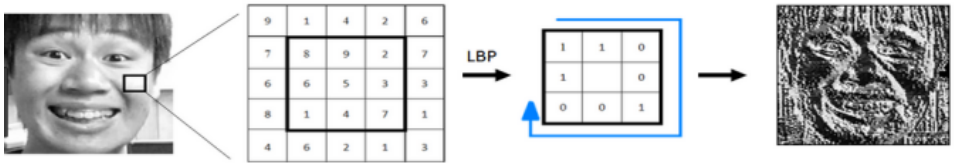


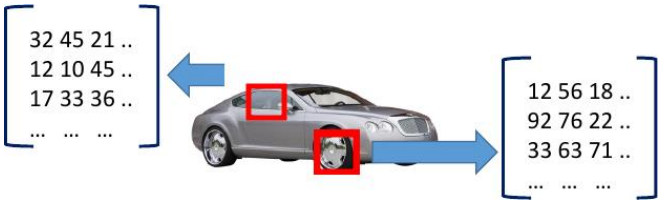
Feature Extraction

- A feature descriptor translates high-dimensional data to a a low dimension feature space
- A feature vector represents the input data produced by the feature descriptor
- Later, a machine learning model will learn the representations



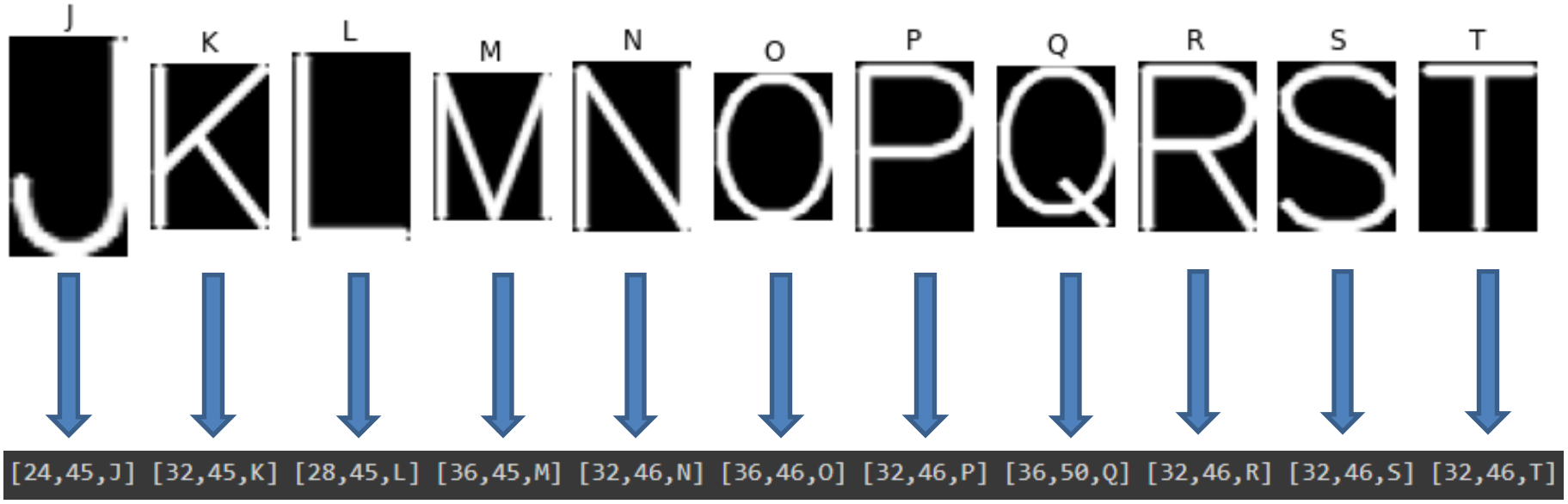






Feature Extraction

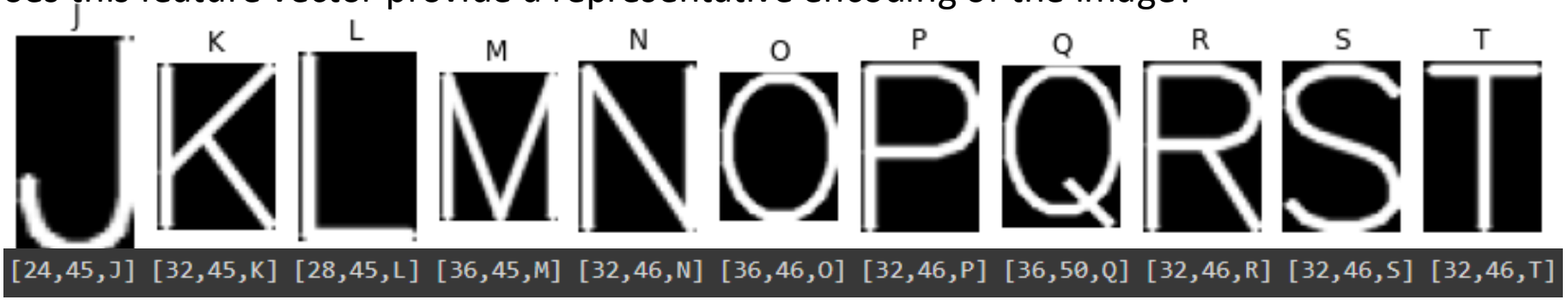
- Let us represent an image by its dimensions. Thus, an image ‘I’ belonging to class ‘X’ can be represented as:
  - f(I,X) = [I.width,I.heigth,X]

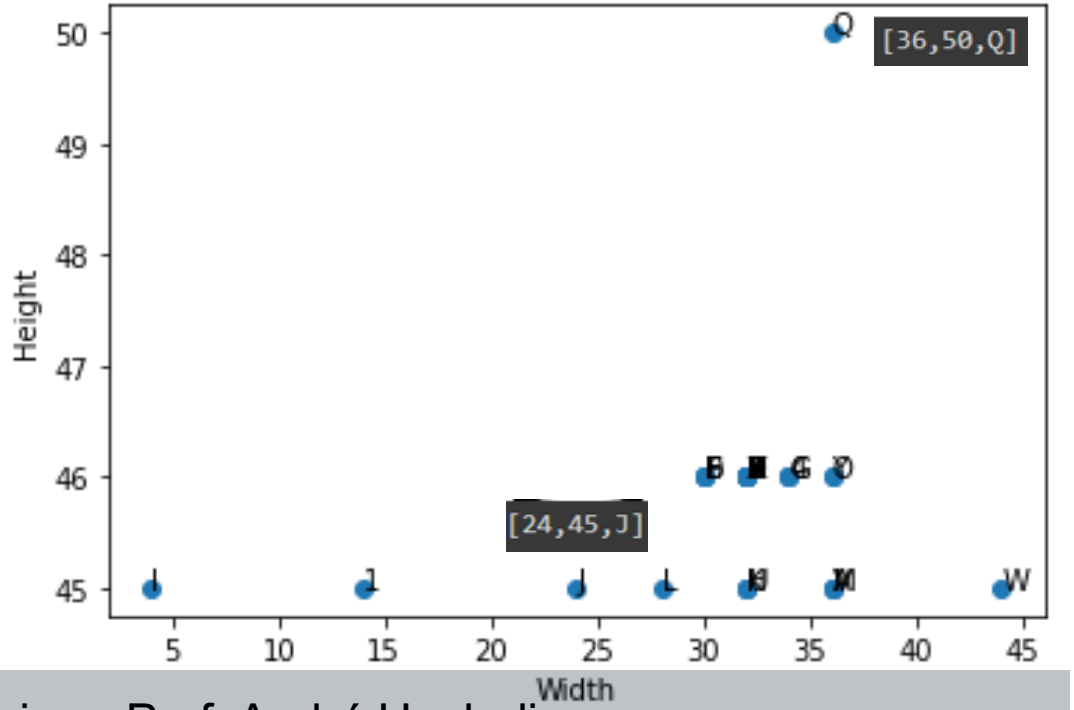


LET’S CODE: Lecture_05_Feature_Extraction_Projections.ipynb

Feature Vector

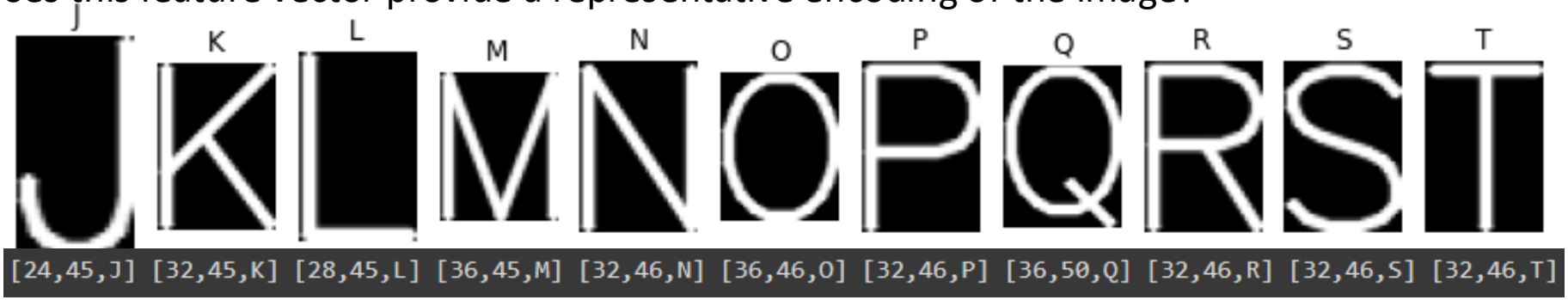
- Does this feature vector provide a representative encoding of the image?

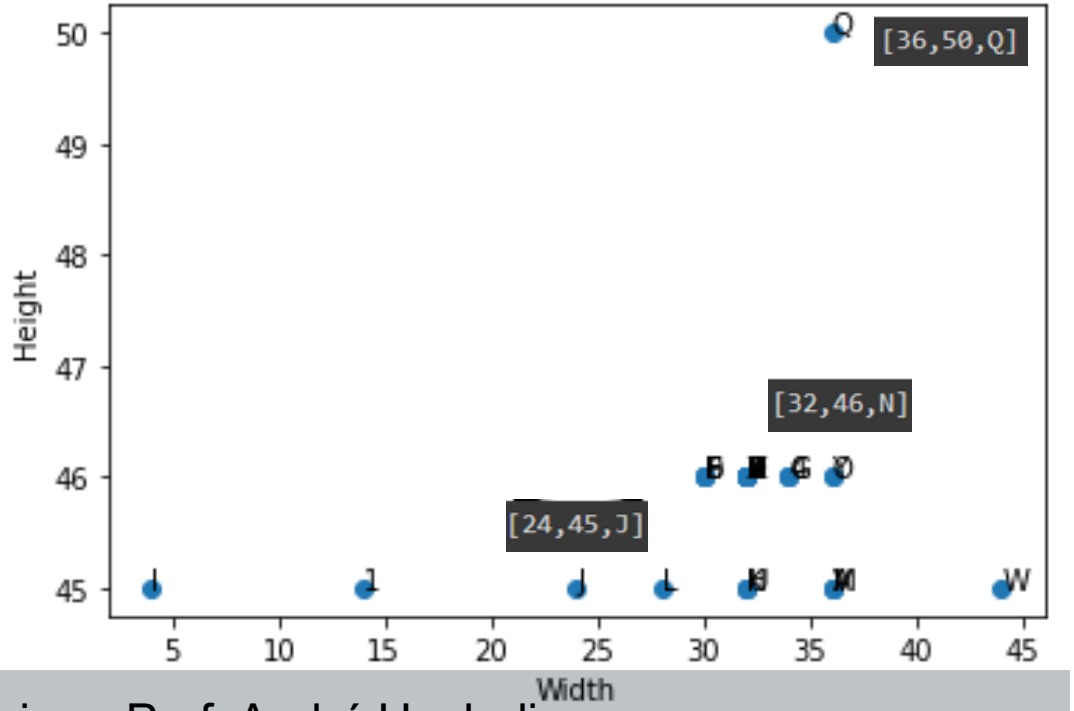




Feature Vector

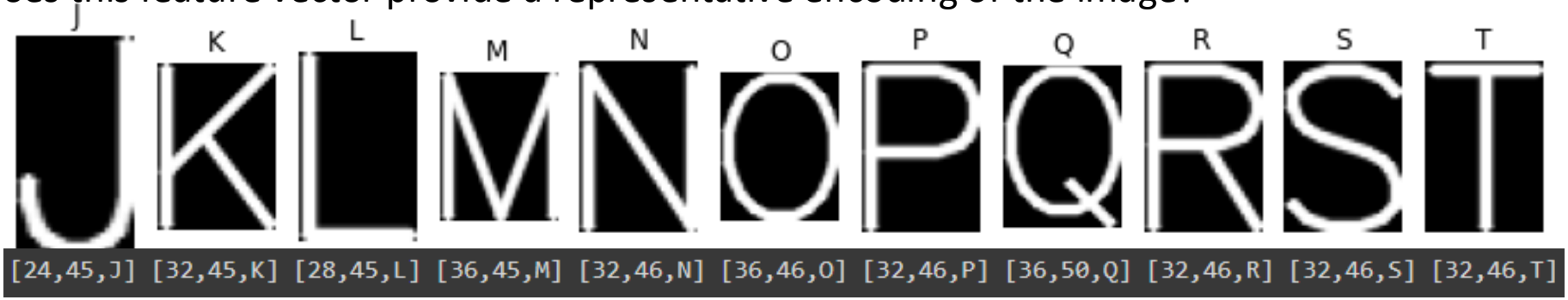
- Does this feature vector provide a representative encoding of the image?

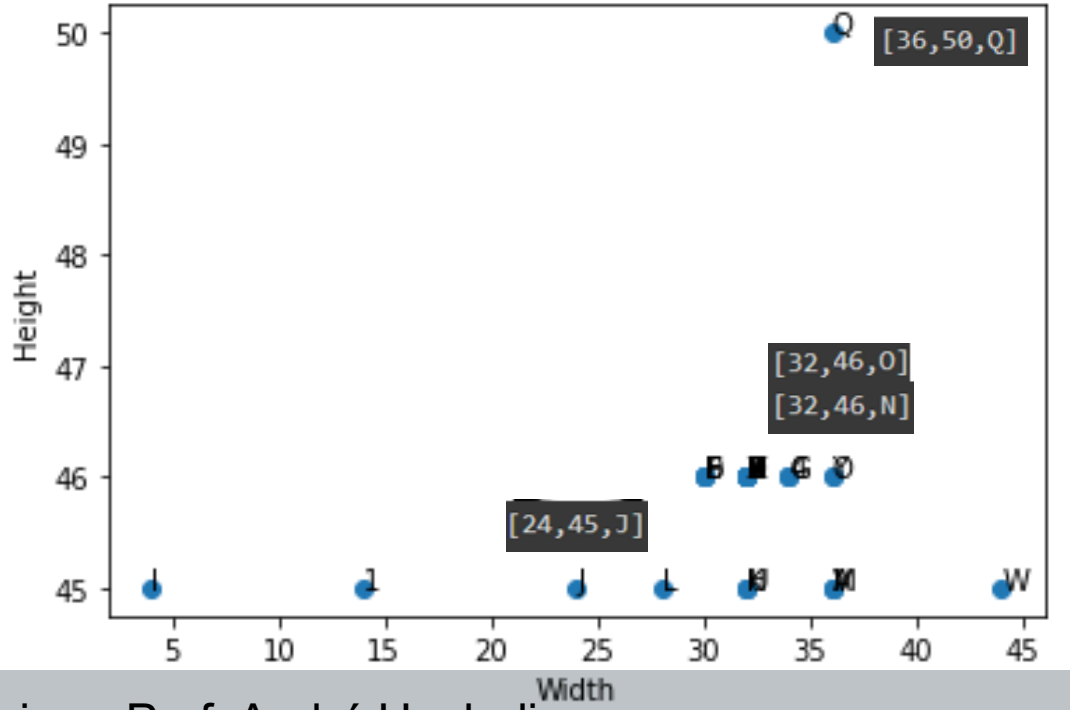




Feature Vector

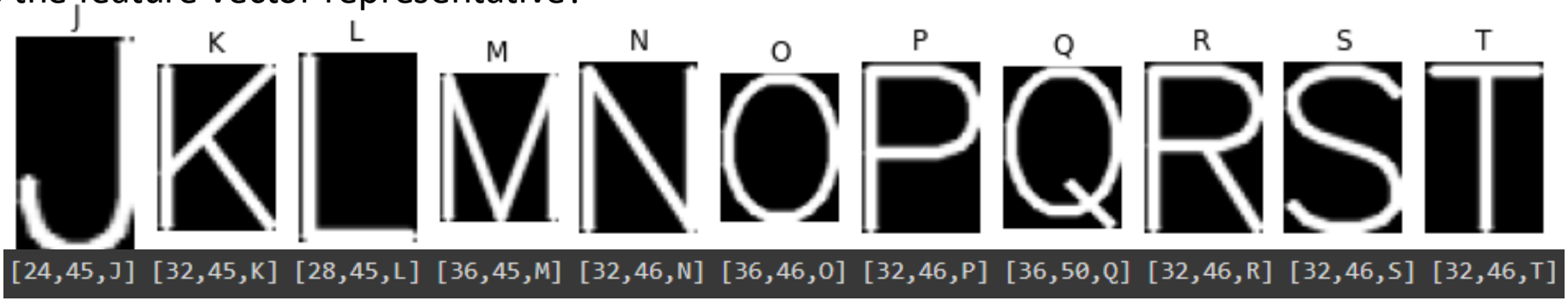
- Does this feature vector provide a representative encoding of the image?

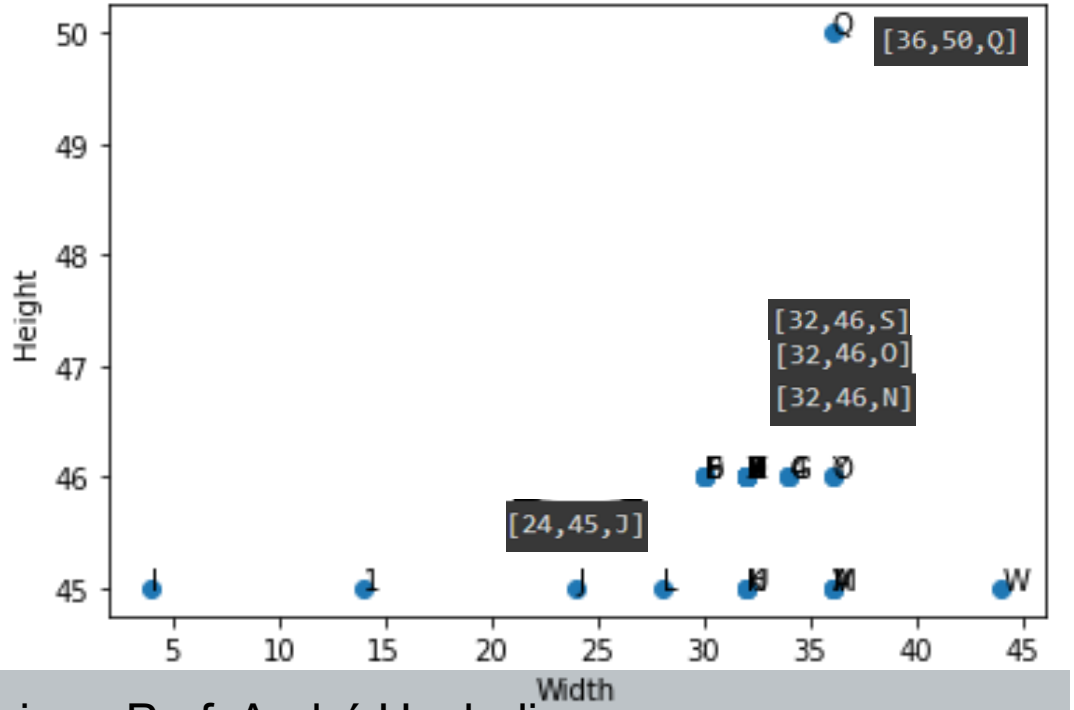




Feature Vector

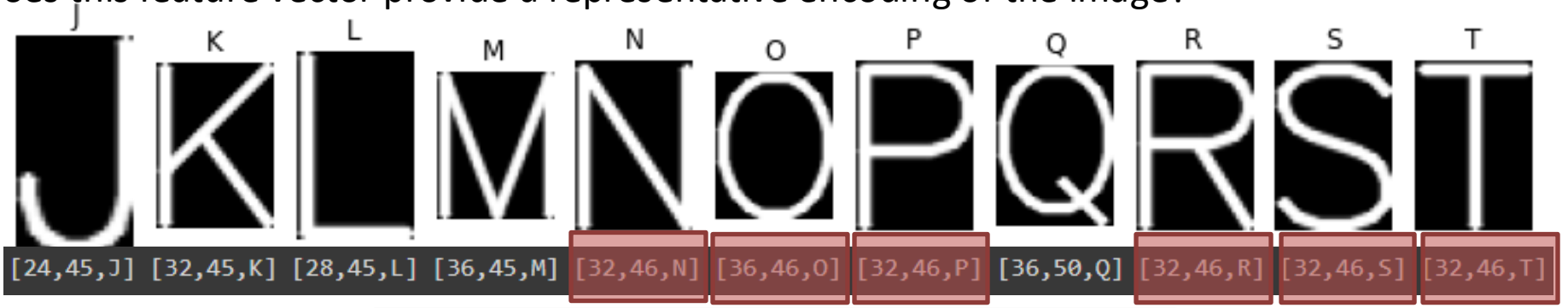
- Is the feature vector representative?

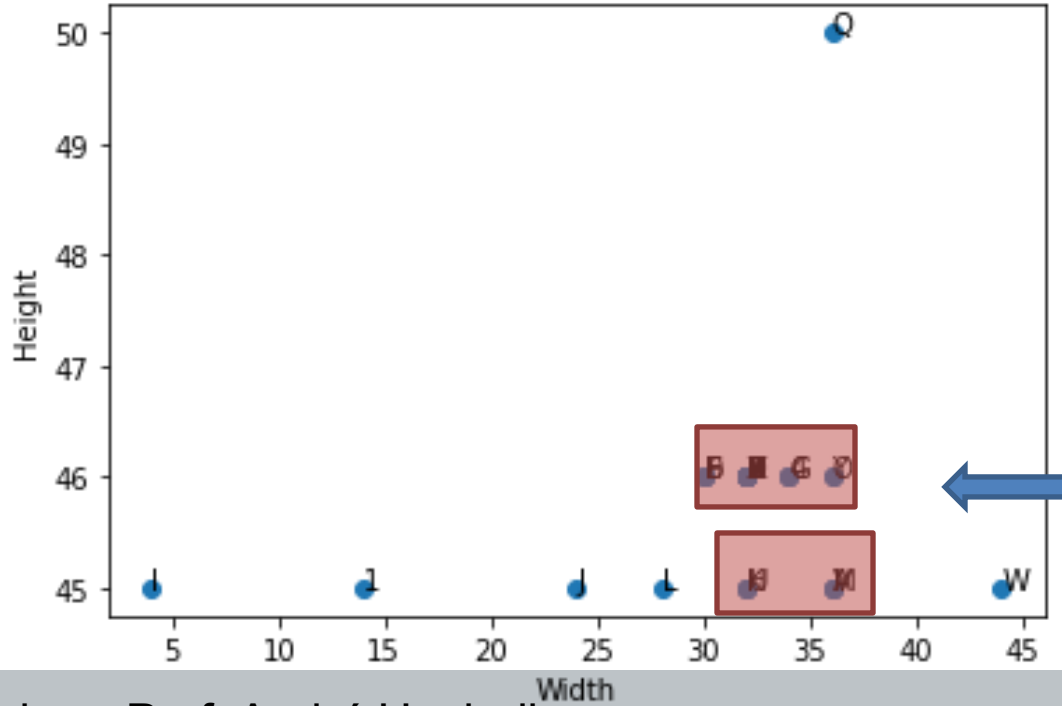




Feature Vector

- Does this feature vector provide a representative encoding of the image?

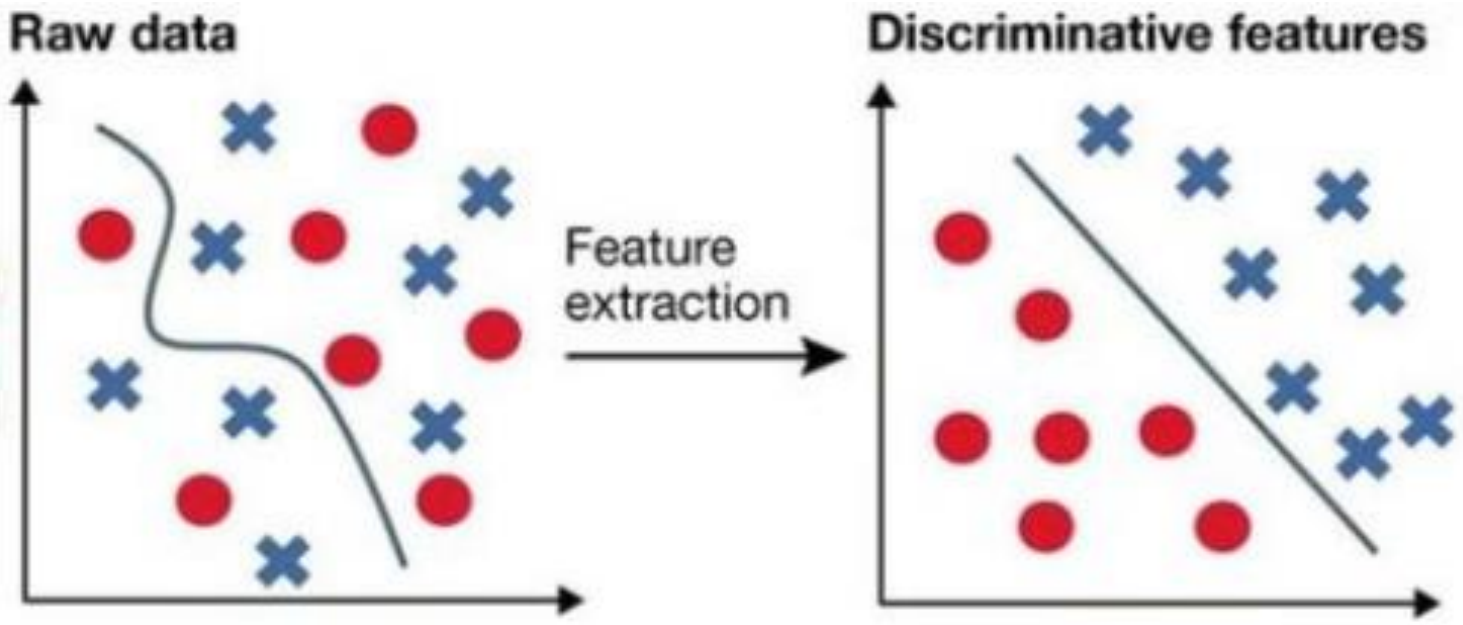




Non Discriminative

Feature Engineering

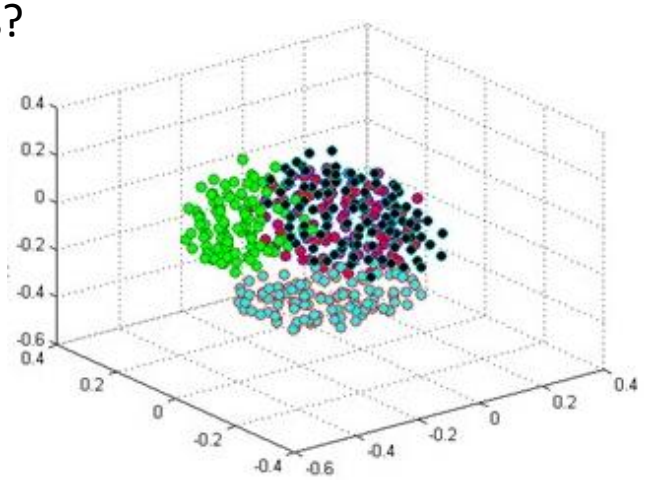
- How to produce a discriminative feature space?
- Features must describe a singular characteristic of the problem for good generalization.



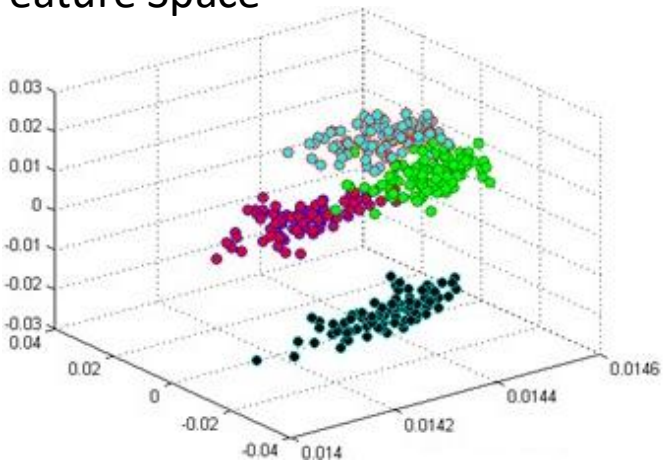
Problem

- How discriminating are features?

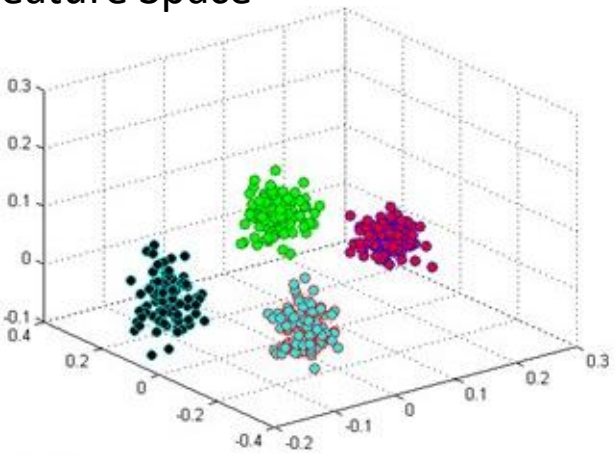
Input Space



Feature Space'



Feature Space''



Feature Space '''

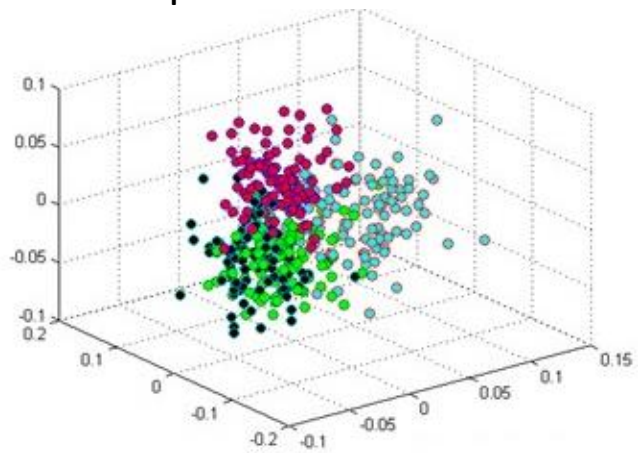
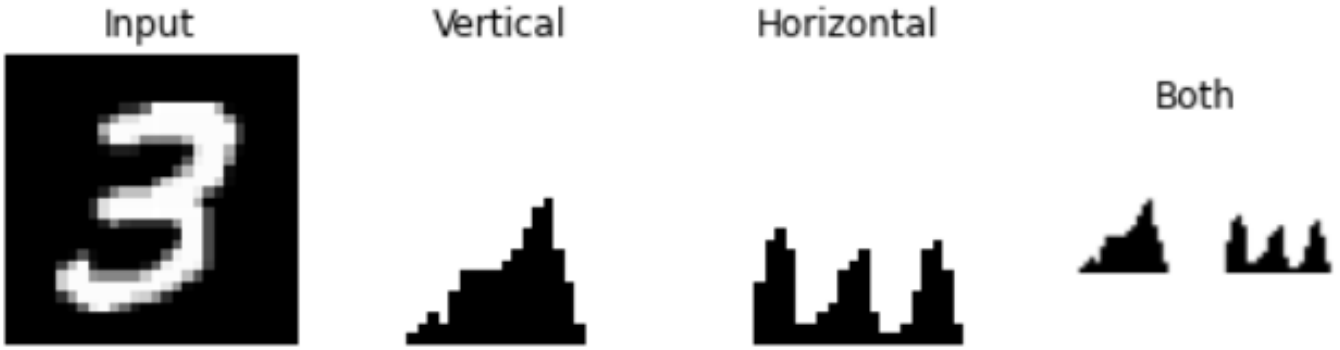


Image Descriptors – Shape/Edges

- Gradient Based
  - Projections



  - Convolutional (Filters)

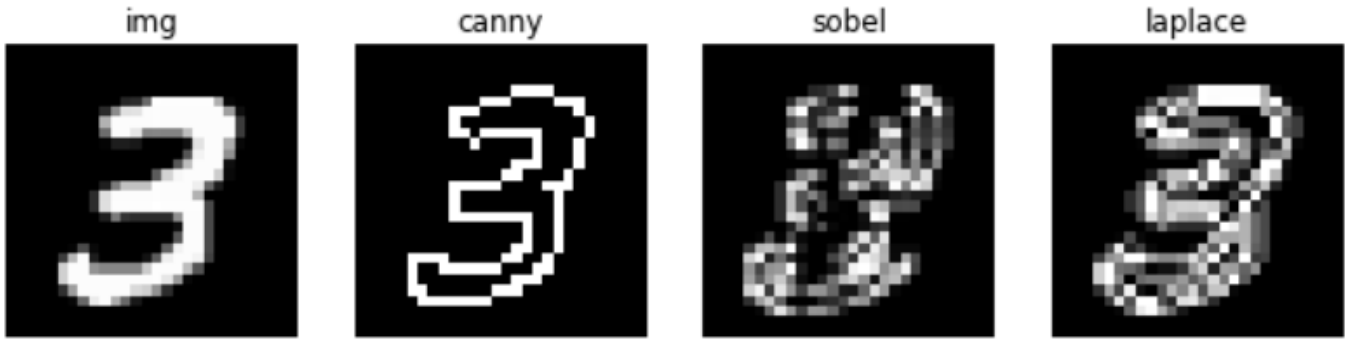
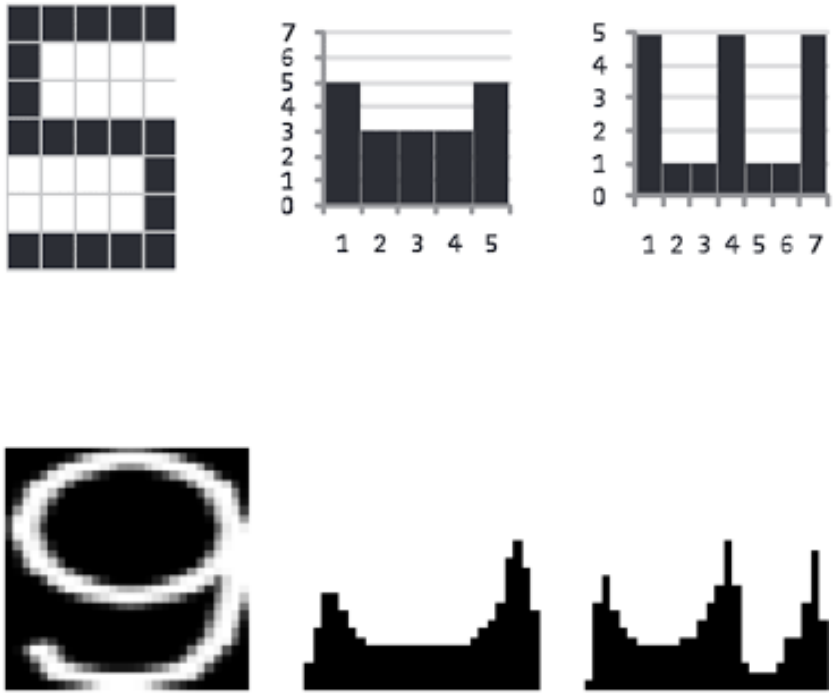
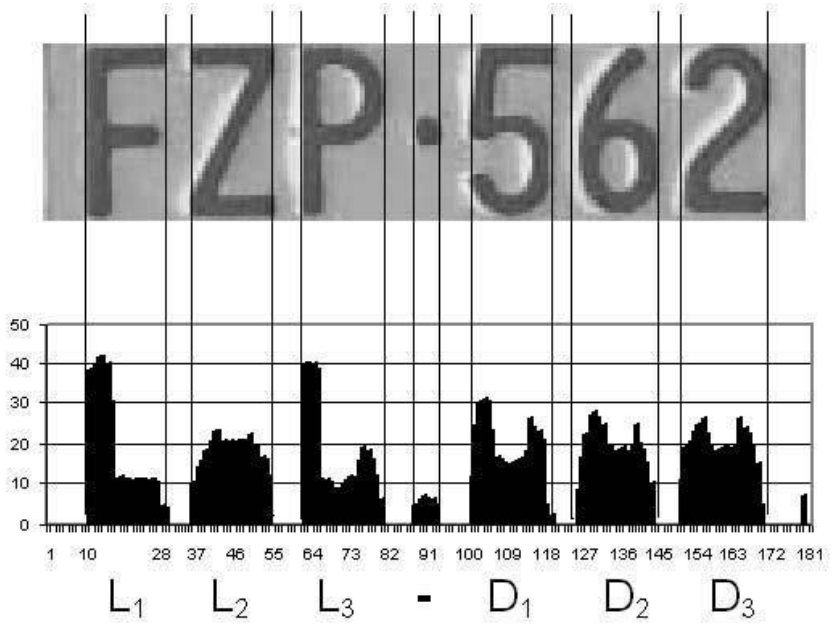


Image Descriptors

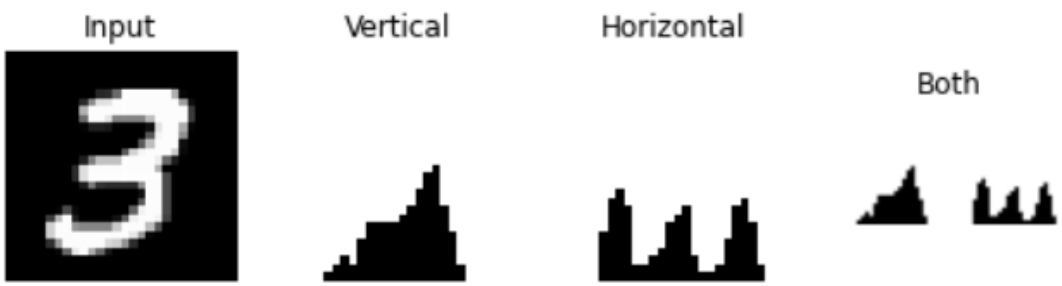
- Vertical and Horizontal Projection





LET'S CODE: [Lecture_05_Feature_Extraction_Projections.ipynb]

Image Descriptors – Shape



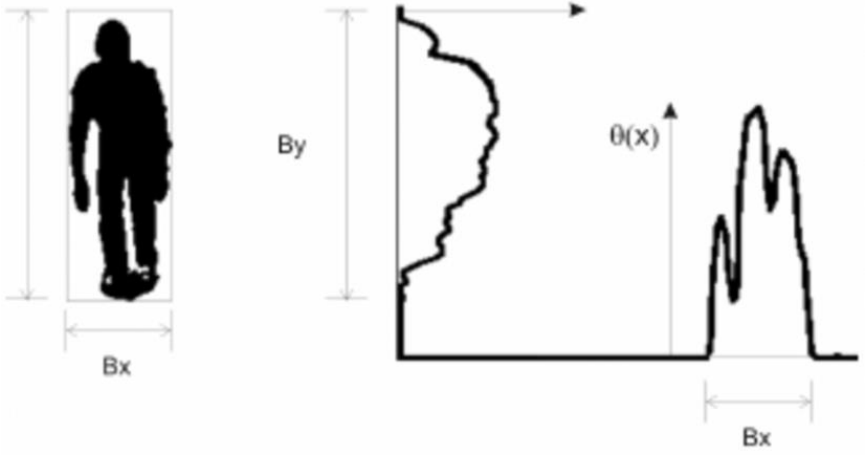
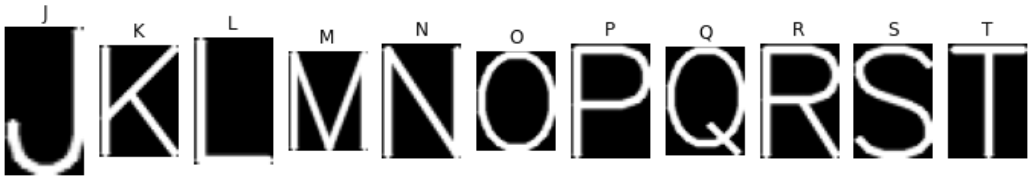
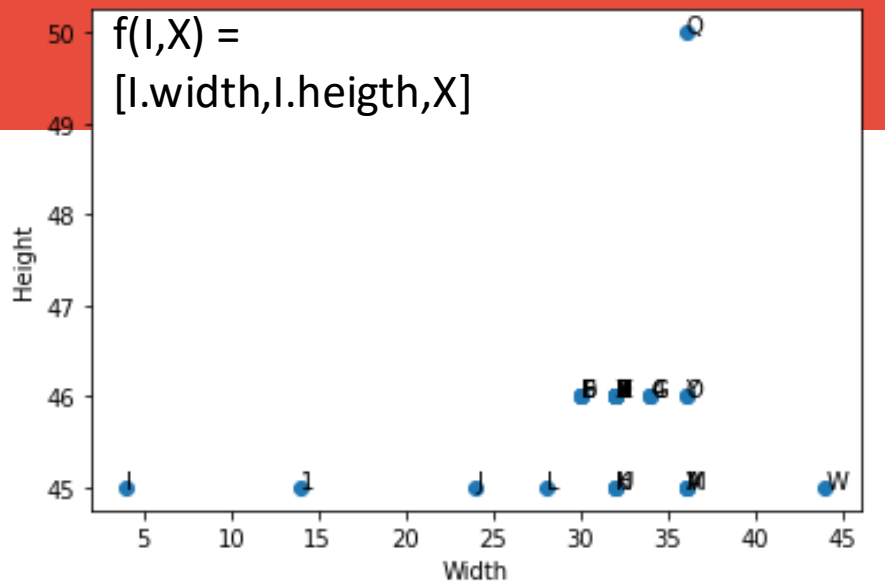
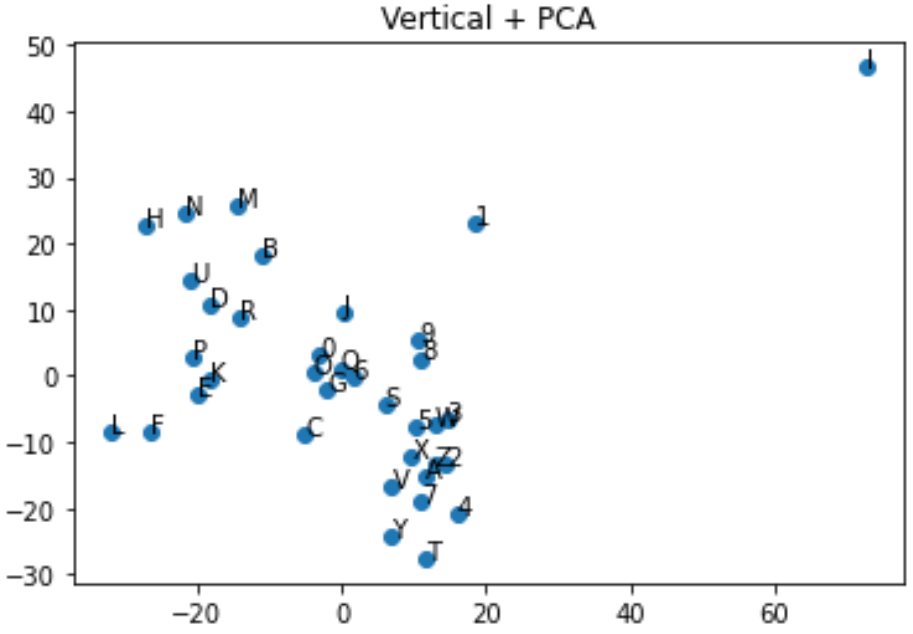


Image Descriptors

- Vertical and Horizontal Projection







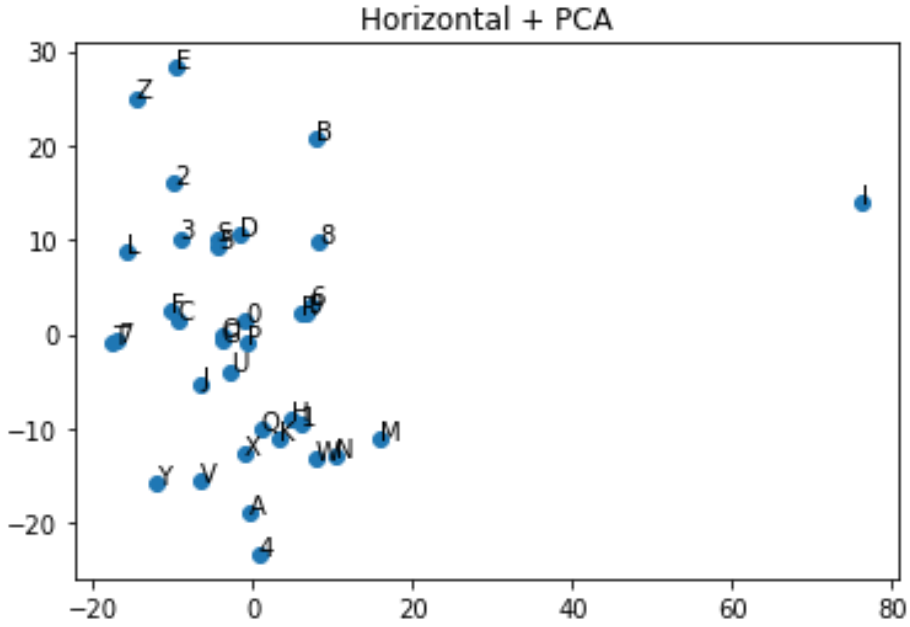


Image Descriptors – Edges

Let's Code: Lecture_05_Image_Descriptors_Edges.ipynb

- Sobel Filter

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

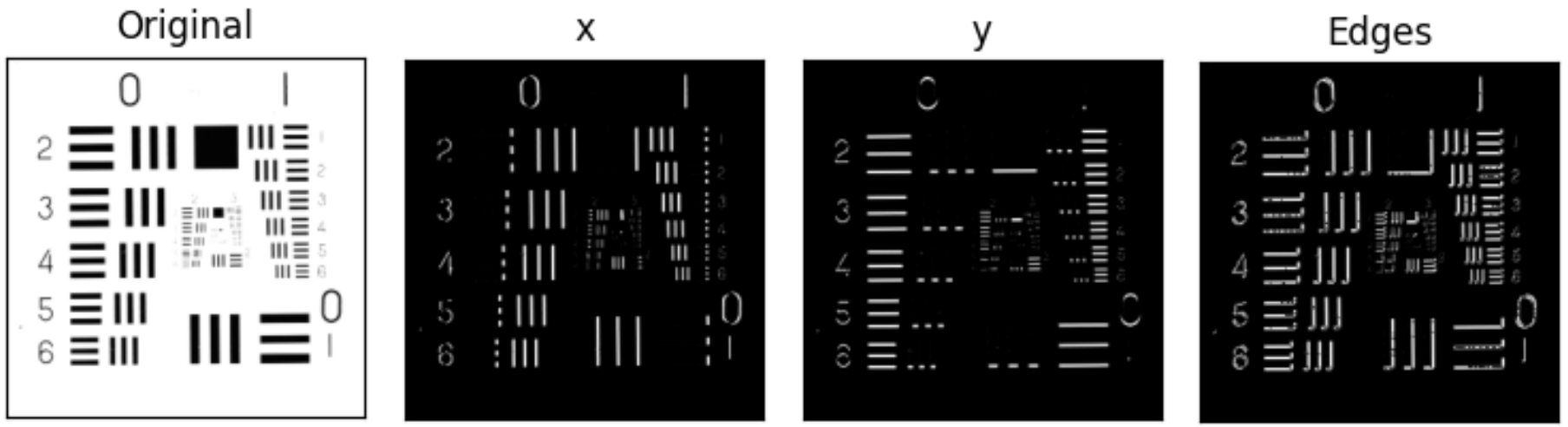


Image Descriptors – Edges

- Sobel Filter

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

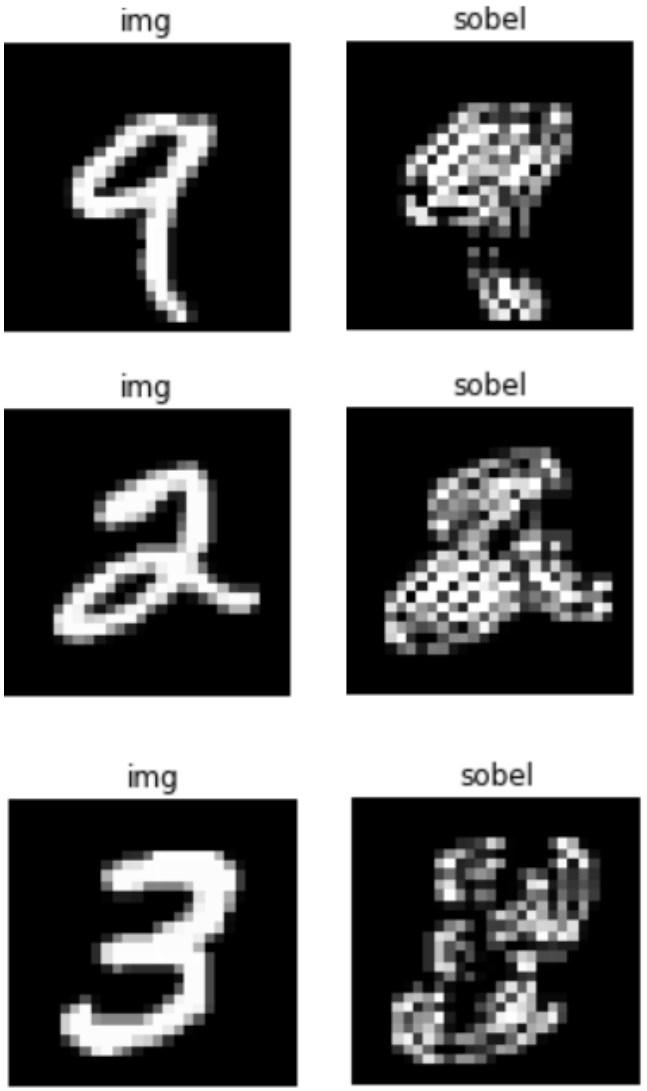


Image Descriptors – Edges

- Laplace

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

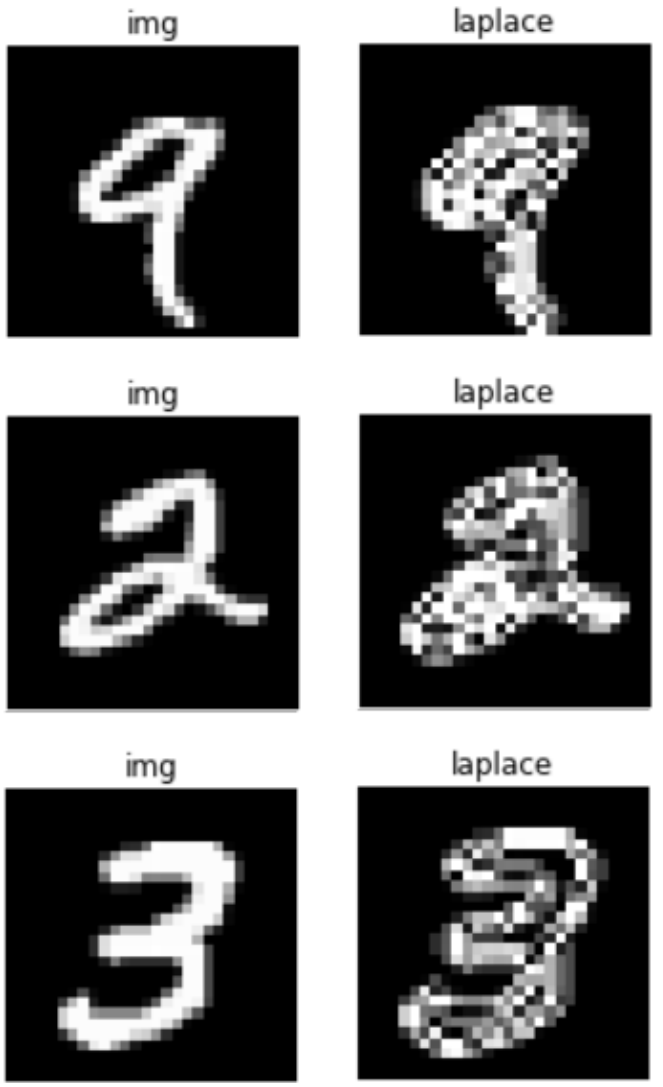


Image Descriptors – Edges

- Canny (John F. Canny 1986)
  - Gaussian Gradient Based Filter
  - Gaussian Blur
  - Gradient Detection

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

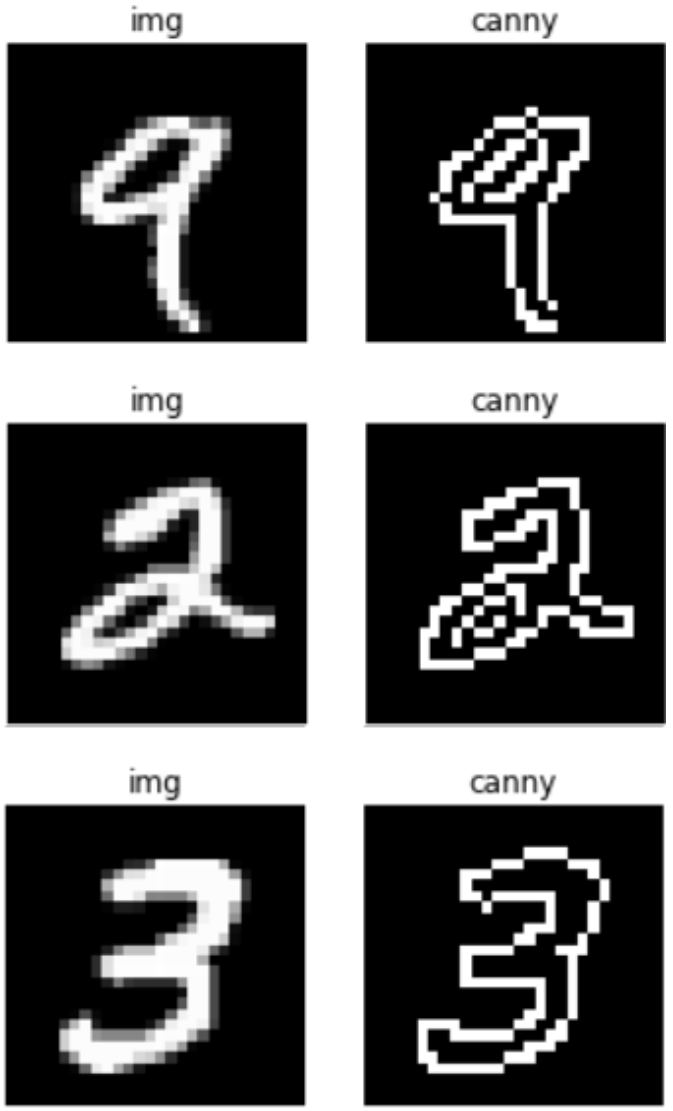


Image Descriptors – Edges

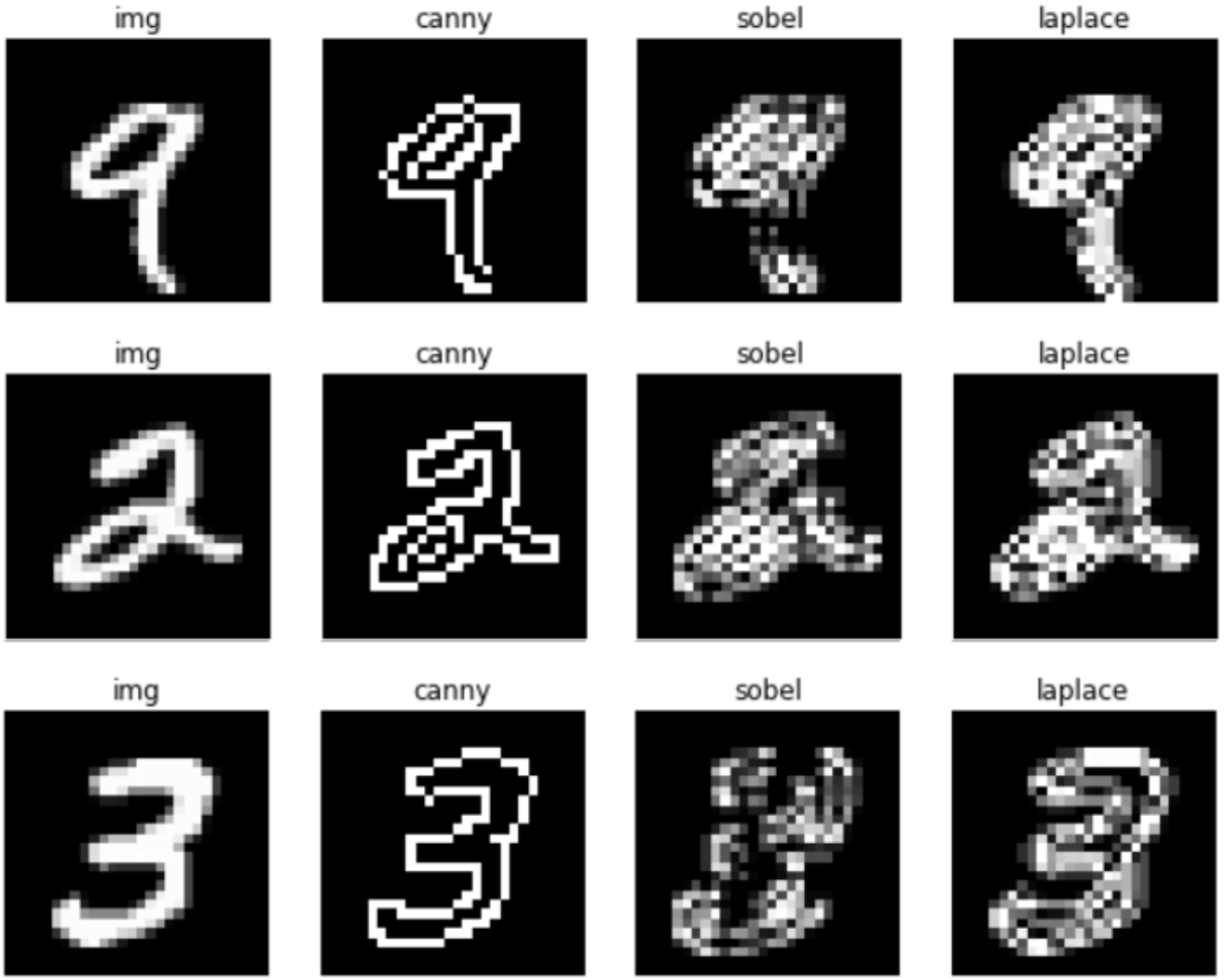
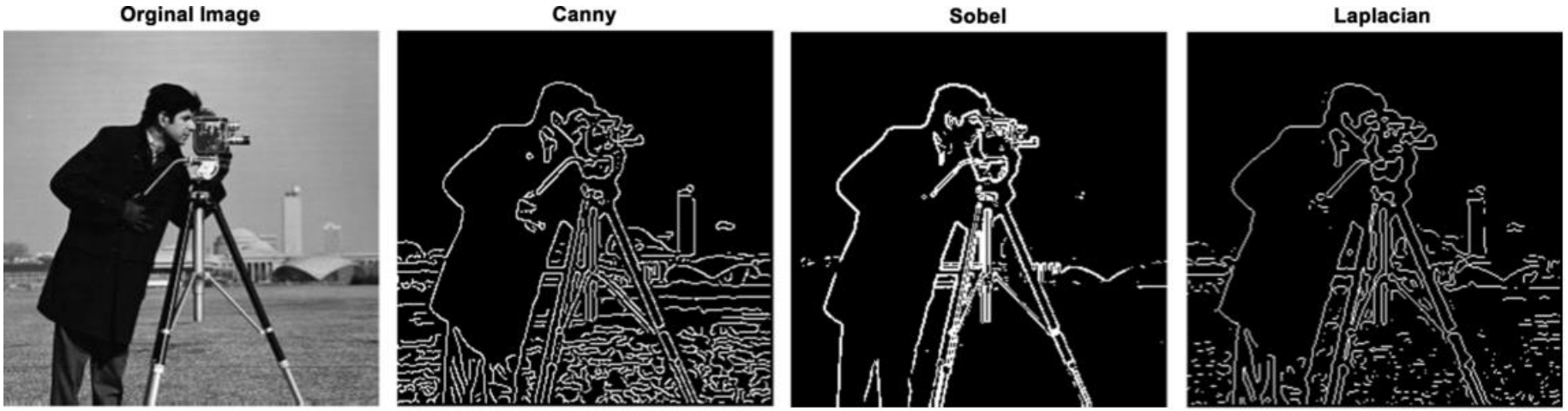
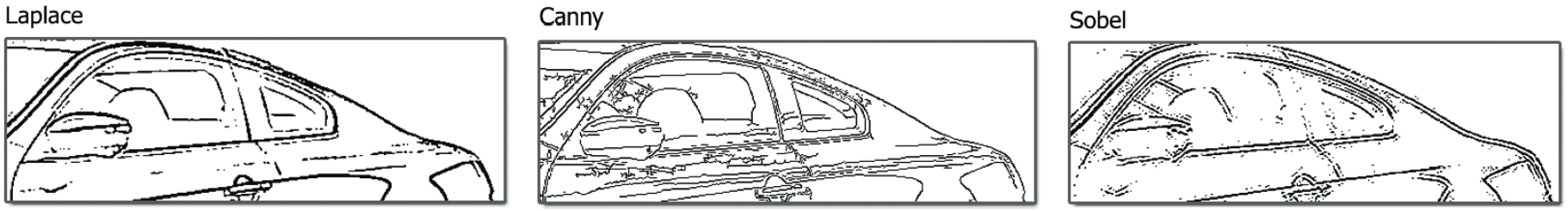


Image Descriptors – Edges





Let's Code: [Lecture_05_Image_Descriptors_Edges.ipynb]

Image Descriptors – Shape

Let's Code: Lecture_05_Image_Descriptors_Texture_and_Others.ipynb

- Moments
  - Values that carry both spatial and intensity information (shape)
    - Weighted average of all pixel's intensities
    - I(x,y) ➔ pixel coordinates of input
    - Powers, *p* and *q,* are the weights of the horizontal and vertical dimensions

- HuMoments (Hu 1962)
  - Translation and Scale Invariant

https://github.com/andrehochuli/teaching/blob/main/ComputerVision/Lecture%2005%20-%20Feature%20Extraction/Lecture_05_Image_Descriptors_Texture_and_Others.ipynb

$$M_{pq} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} x^p y^q I(x,y)$$

$$h_1 = \eta_{20} + \eta_{02}$$

$$h_2 = (\eta_{20} - \eta_{02})^2 + 4(\eta_{11})^2$$

$$h_3 = (\eta_{30} - 3\eta_{12})^2 + 3(\eta_{03} - 3\eta_{21})^2$$

$$h_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2$$

$$h_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + (3\eta_{21} - \eta_{03})(\eta_{03} + \eta_{21})[3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]$$

$$h_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - 7(\eta_{03} + \eta_{21})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})$$

$$h_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + (\eta_{30} - 3\eta_{12})(\eta_{03} + \eta_{21})[3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]$$

Image Descriptors – Shape

- HoG – Histogram of Oriented Gradients
  - Computes the gradient and orientation of edges
  - Use a kernel to compute the Gradients (i.e 9x1)
  - Patch-Based Histogram (8x8, 16x16..)

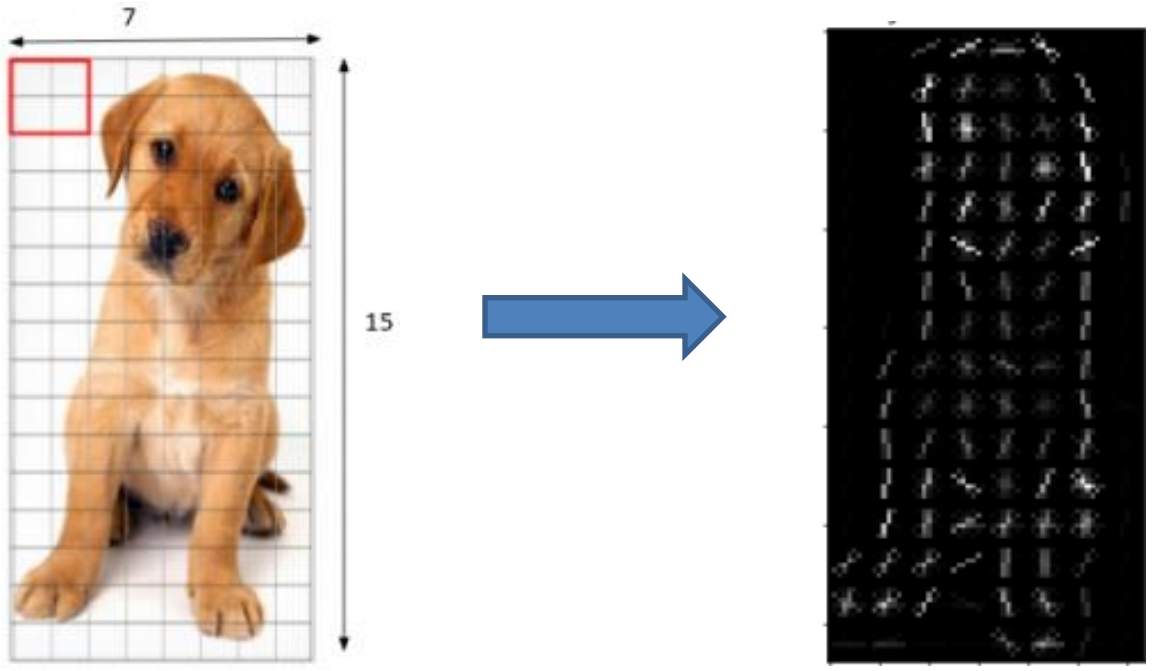
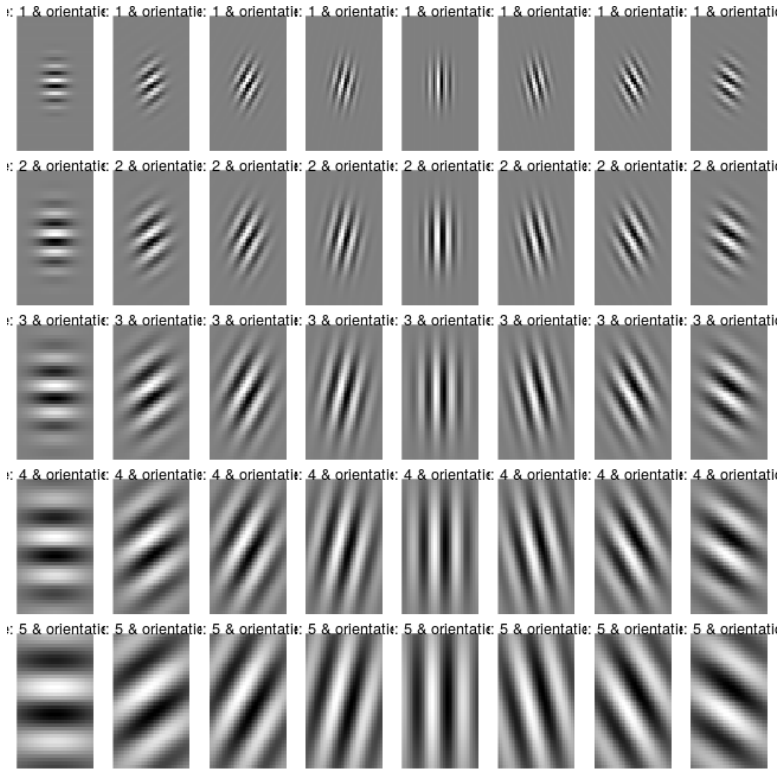


Image Descriptors – Texture

- Gabor Filters
  - Convolves the image using several Gaussian Kernels (Kernel Bank)



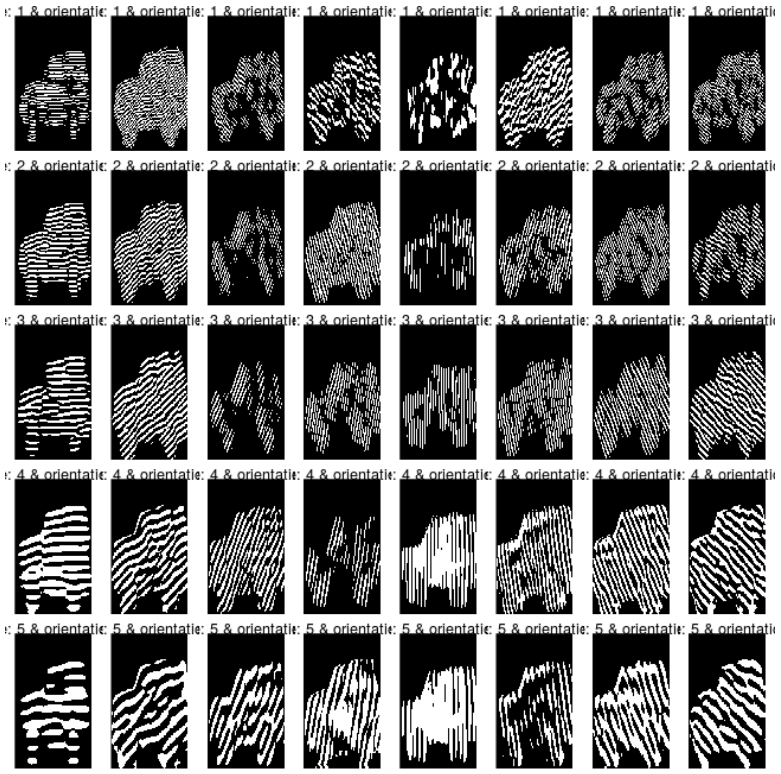
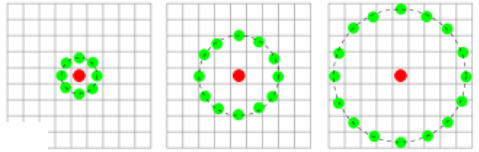
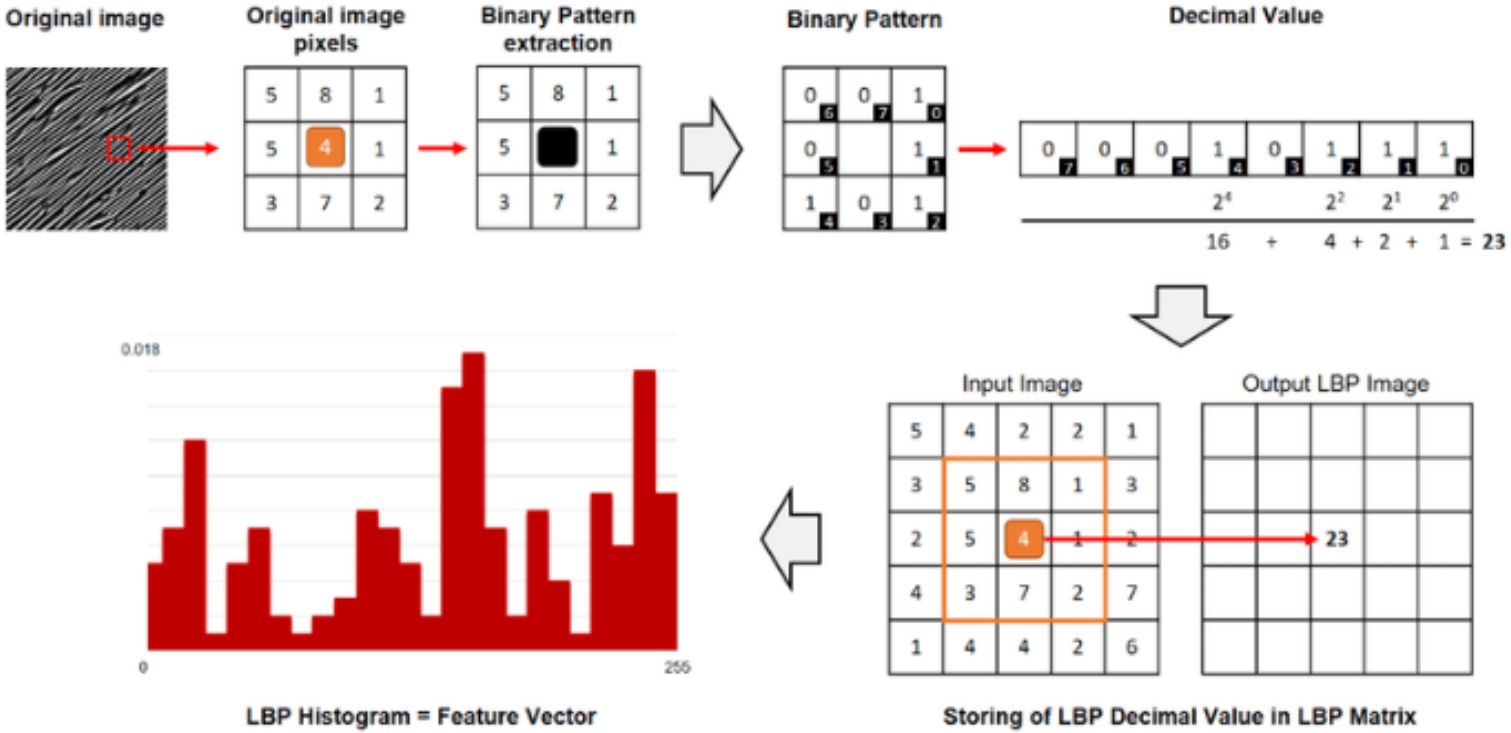


Image Descriptors – Texture

- Local Binary Patterns
  - Convolves the image using a Circular Kernel
  - The resulting pixel is computed in the binary neighborhood





Let's Code: [Lecture_05_Image_Descriptors_Texture_and_Others.ipynb]