BE/Bi 205: Lecture 2

David Van Valen MD, PhD

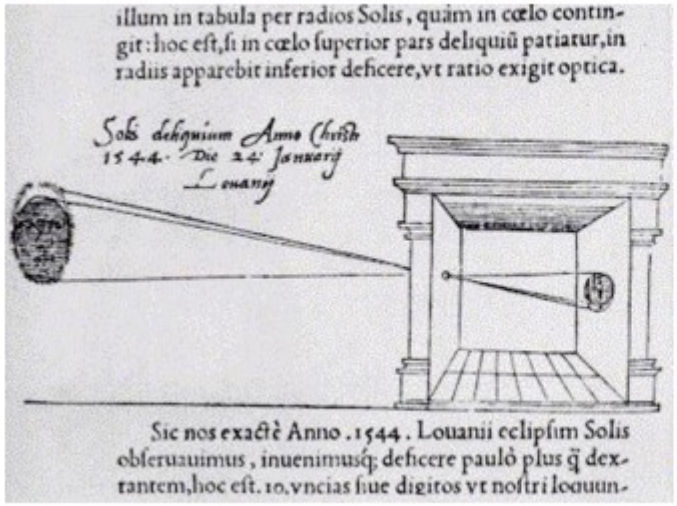
04/06/2023

Today's Lecture

- Images as data
- Sequences as data
- Structures as data

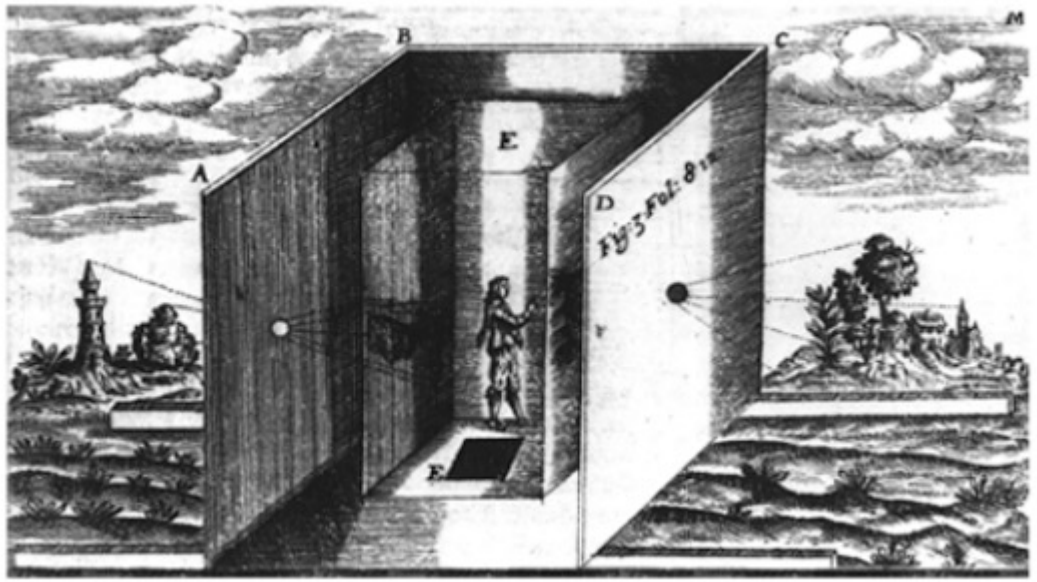
What is an image?

- A visual representation in the form of a function f(x, y) where f describes brightness (or color) at a point (x, y)
- Continuous in amplitude and space
- Most images are defined over a rectangle (but they don't have to be)


illum in tabula per radios Solis, quàm in cœlo contingit: hoc eſt, ſi in cœlo ſuperior pars deliquiũ patiatur, in radiis apparebit inferior deficere, vt ratio exigit optica.

Solis deliquium Anno Chriſti 1544. Die 24. Januarij Louanij

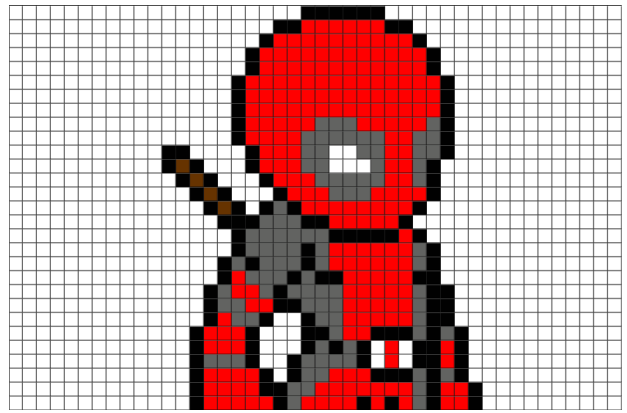
Sic nos exactè Anno .1544. Louanii eclipſim Solis obſeruauimus, inuenimusq; deficere paulò plus q̃ dextantem, hoc eſt. 10. vncias ſiue digitos vt noſtri loquun-



Athanasius Kircher, Dark Chamber with Lenses, 1946

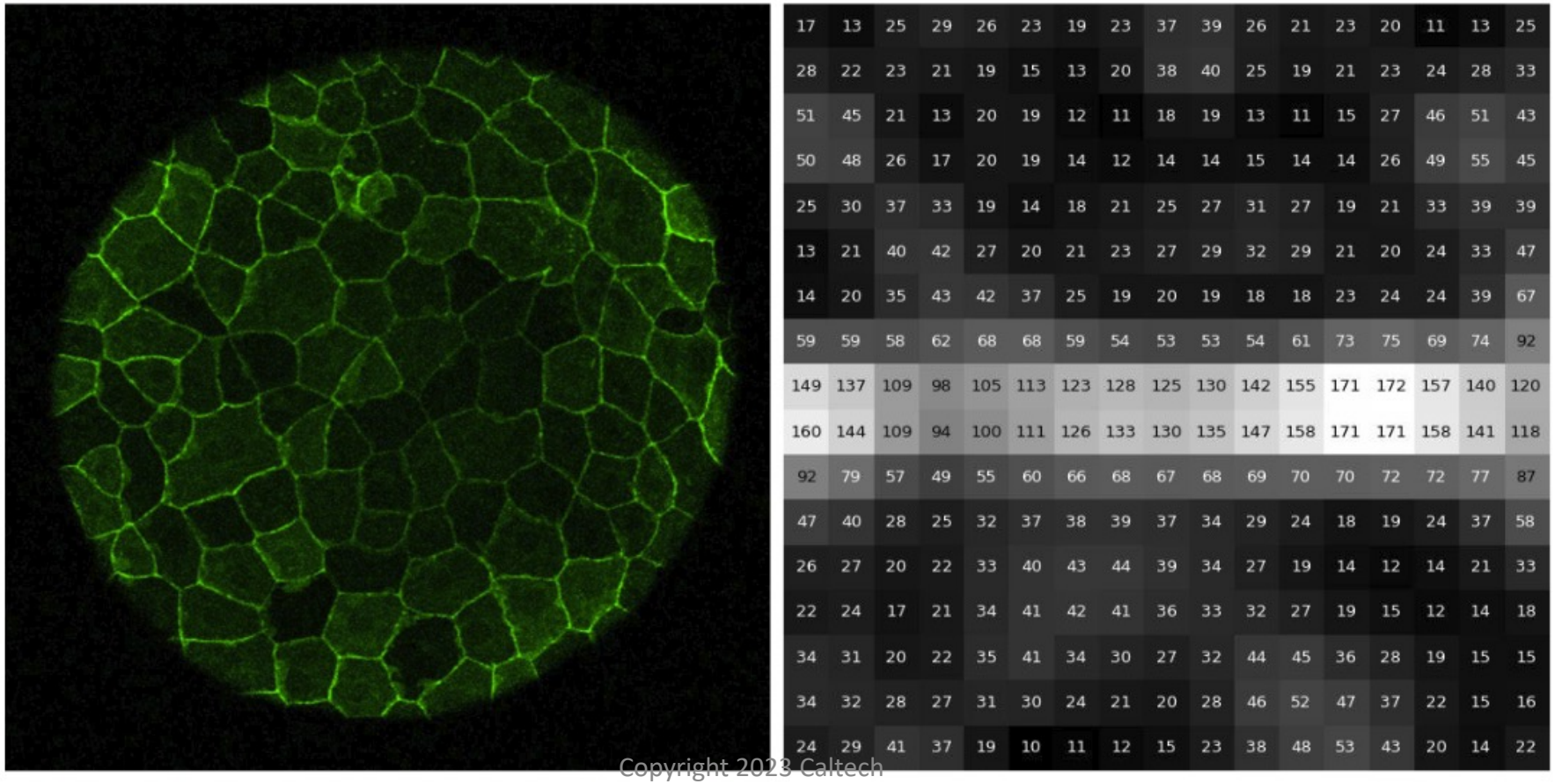
What is a digital image?

- Discrete samples f[x, y] representing the continuous image f(x, y)
- Each element in this 2D array f[x, y] is called a pixel (from picture element)



What is a digital image?

- Ultimately a digital image is an array of numbers

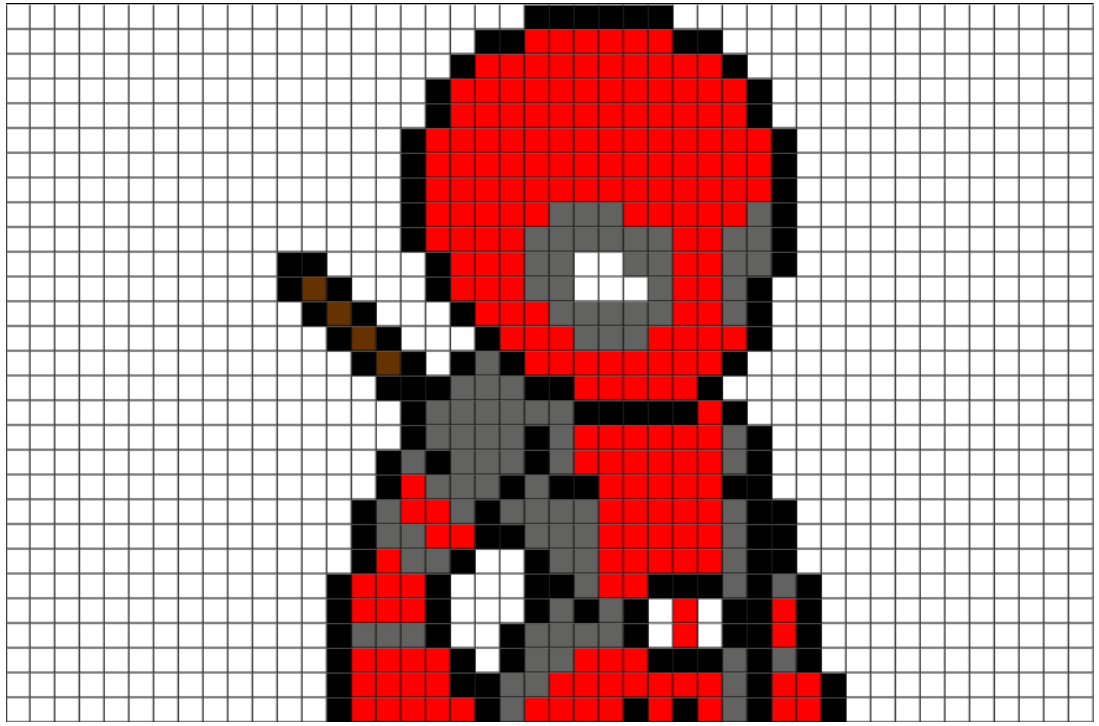


17	13	25	29	26	23	19	23	37	39	26	21	23	20	11	13	25
28	22	23	21	19	15	13	20	38	40	25	19	21	23	24	28	33
51	45	21	13	20	19	12	11	18	19	13	11	15	27	46	51	43
50	48	26	17	20	19	14	12	14	14	15	14	14	26	49	55	45
25	30	37	33	19	14	18	21	25	27	31	27	19	21	33	39	39
13	21	40	42	27	20	21	23	27	29	32	29	21	20	24	33	47
14	20	35	43	42	37	25	19	20	19	18	18	23	24	24	39	67
59	59	58	62	68	68	59	54	53	53	54	61	73	75	69	74	92
149	137	109	98	105	113	123	128	125	130	142	155	171	172	157	140	120
160	144	109	94	100	111	126	133	130	135	147	158	171	171	158	141	118
92	79	57	49	55	60	66	68	67	68	69	70	70	72	72	77	87
47	40	28	25	32	37	38	39	37	34	29	24	18	19	24	37	58
26	27	20	22	33	40	43	44	39	34	27	19	14	12	14	21	33
22	24	17	21	34	41	42	41	36	33	32	27	19	15	12	14	18
34	31	20	22	35	41	34	30	27	32	44	45	36	28	19	15	15
34	32	28	27	31	30	24	21	20	28	46	52	47	37	22	15	16
24	29	41	37	19	10	11	12	15	23	38	48	53	43	20	14	22

Jonas Hartmann, Computational Image Analysis, 2017

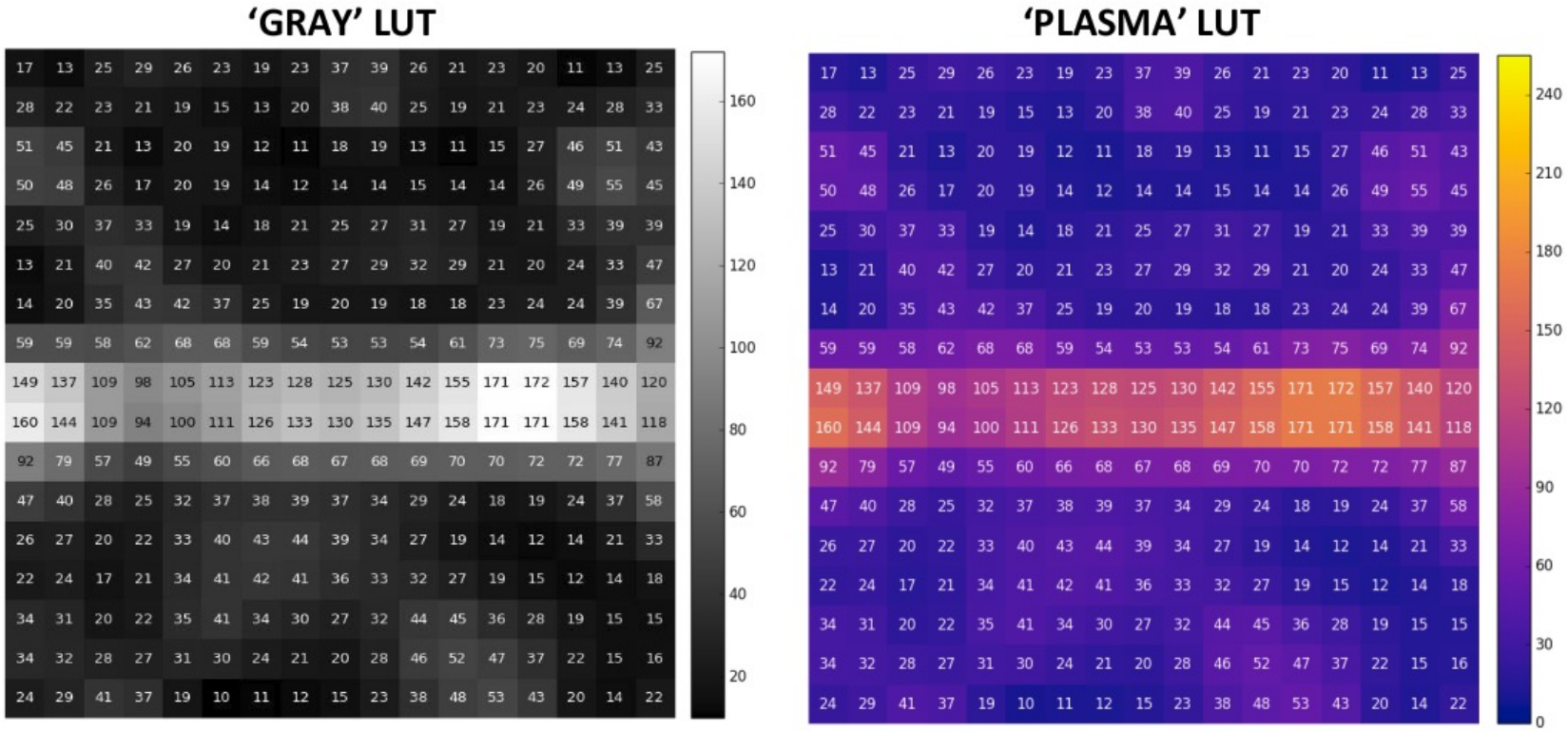
Useful digital image properties

- Spatial resolution
  - Width and height in pixels/length (inches, cm, etc.)
- Intensity
- Number of channels
  - RGB is 3 channels, grayscale is 1
  - Alpha channel is used to indicate transparency
- Color spaces
  - RGB, CMYK, HSV



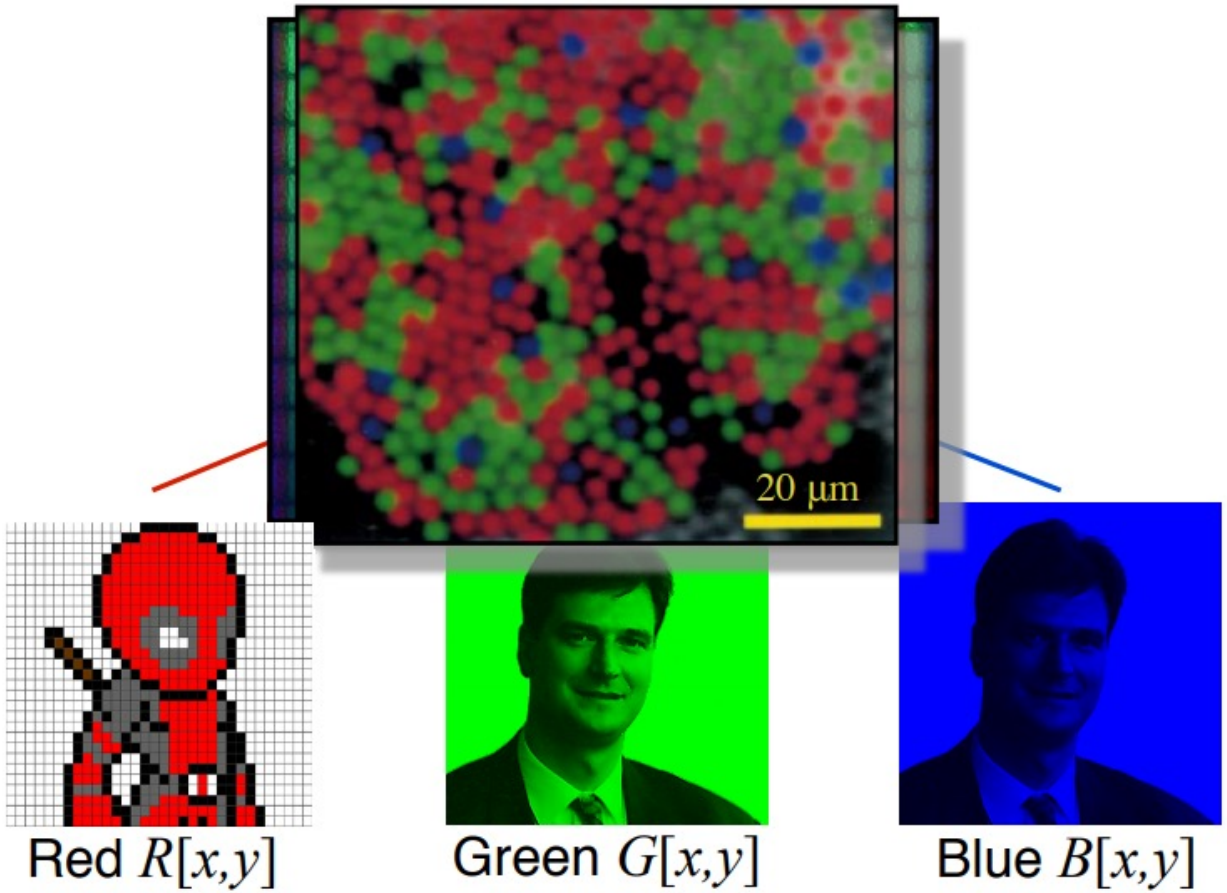
Color components

- The link between array values in single channel images and color images are look up tables (LUTs)



Jonas Hartmann, Computational Image Analysis, 2017

Color comoponents



Monochrome image

$R[x,y] = G[x,y] = B[x,y]$

Digital images are multi-dimensional arrays

- Multi-channel images, time series images, and image stacks are just multi-dimensional arrays

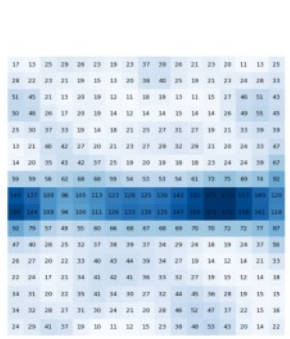
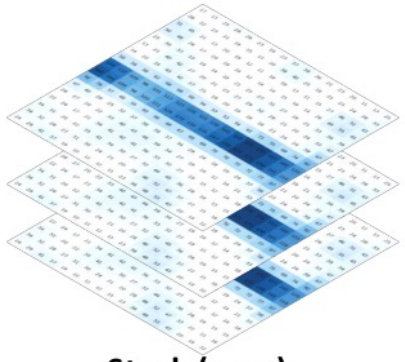
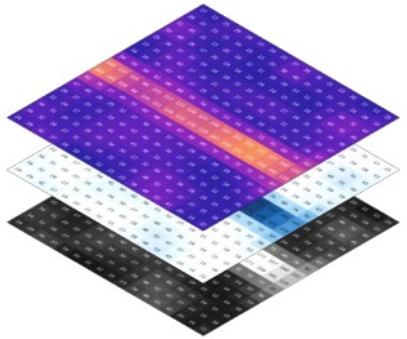


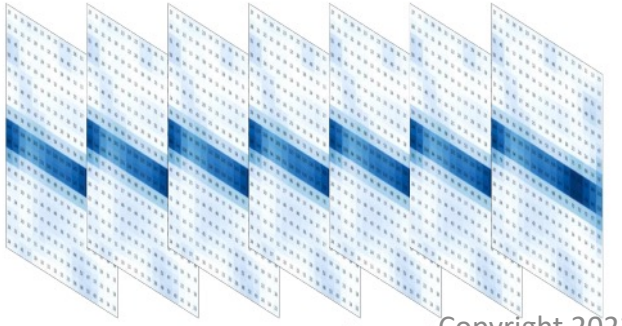
Image (y,x)



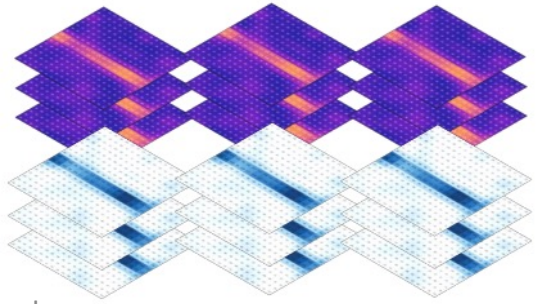
Stack (z,y,x)



Channels (c,y,x)



Time Series (t,y,x)



(t,c,z,y,x)

Jonas Hartmann, Computational Image Analysis, 2017

Digital hangover: data types

- Digital images vary in the precision used to store the array values

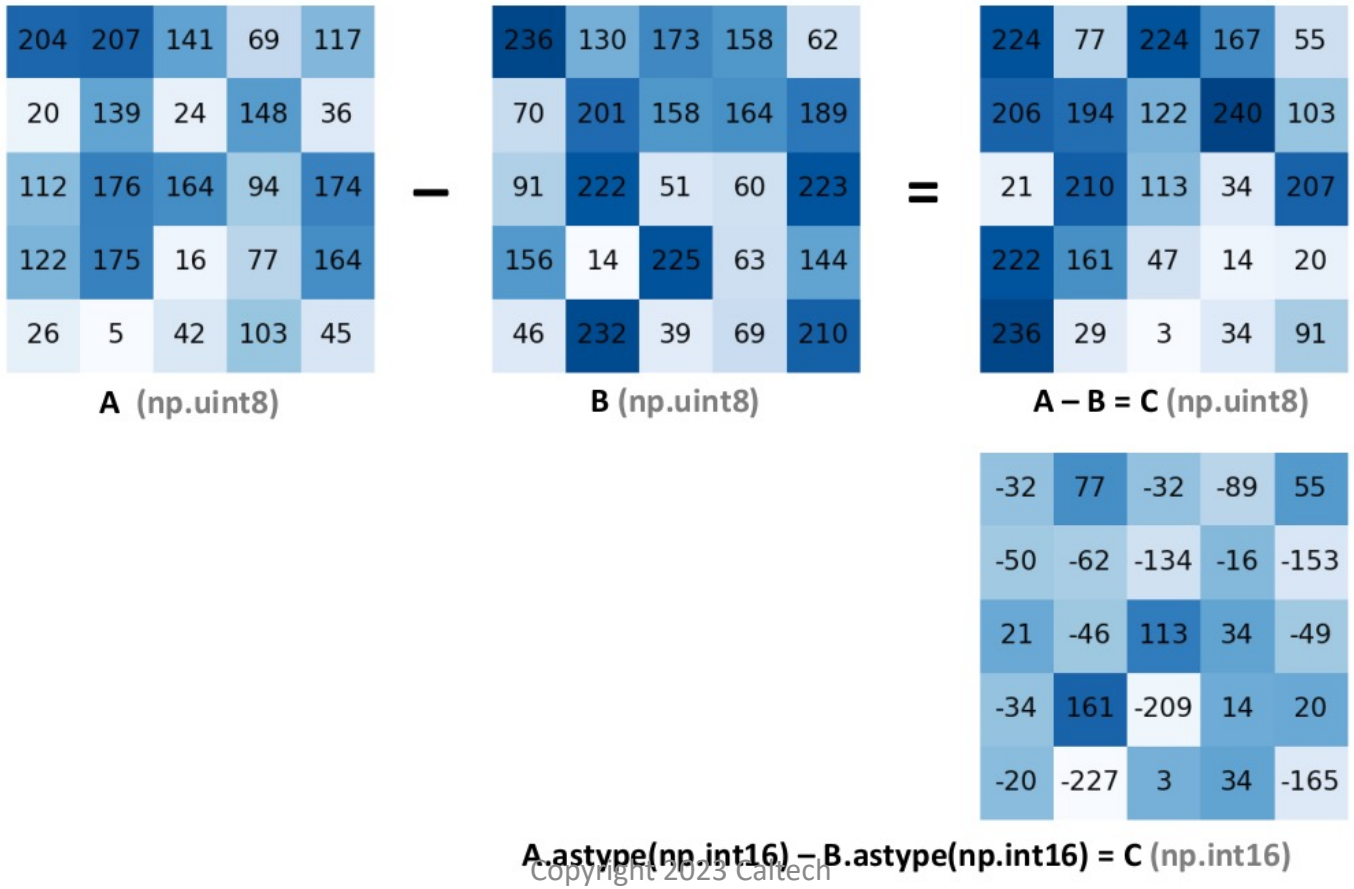
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa

Used in image processing

Produced by microscope

Digital hangover: data types

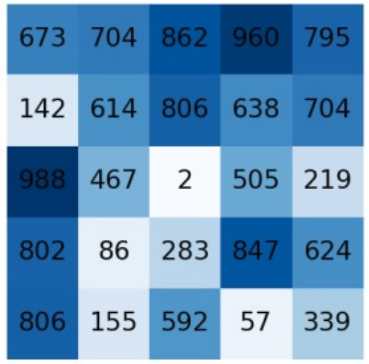
- A cautionary tale – image subtraction



Jonas Hartmann, Computational Image Analysis, 2017

Digital hangover: data types

- A cautionary tale - rescaling



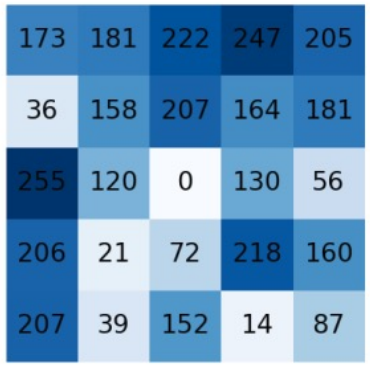
A (np.uint16)

=

A.astype(np.uint8) = C (np.uint8)

Rescales an array between 0 and x:

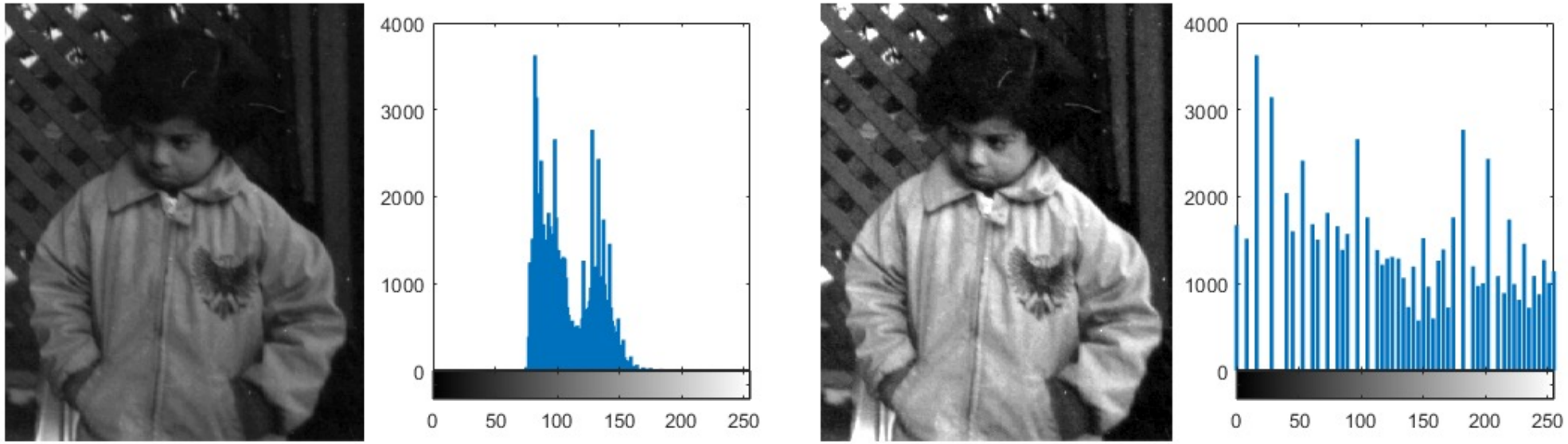
$$\frac{A - min(A)}{max(A) - min(A)} \cdot x = C$$



(((A - A.min()) / (A.max() - A.min()))*255).astype(np.uint8) = C (np.uint8)

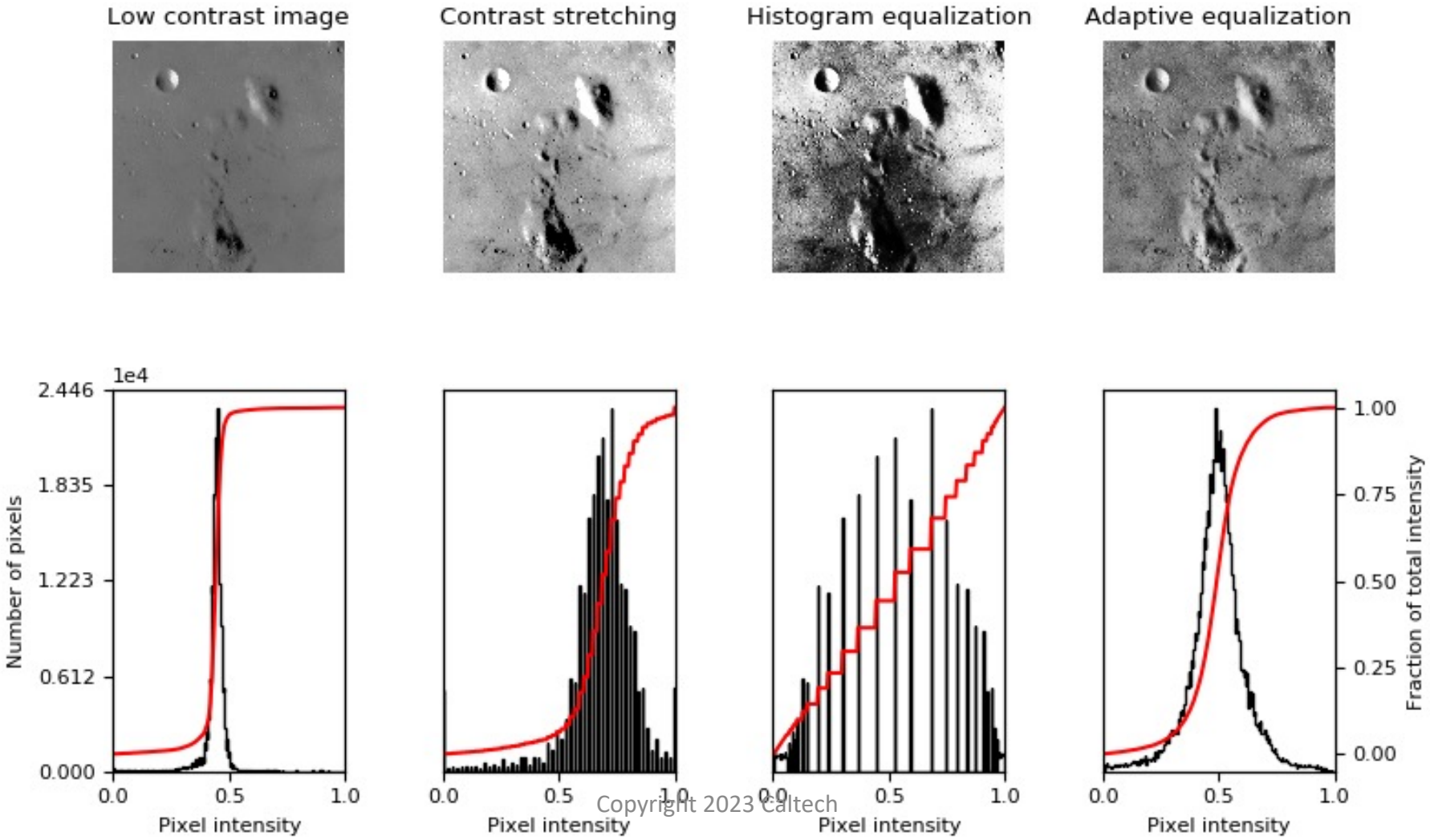
Digital image processing: Contrast adjustment

- Contrast adjustment is the process of changing pixel intensities to take advantage of the full range of values
- The goal is to maximize the intensity differences between different objects or parts of the image, making them easier to see



https://www.mathworks.com/help/images/contrast-adjustment.html

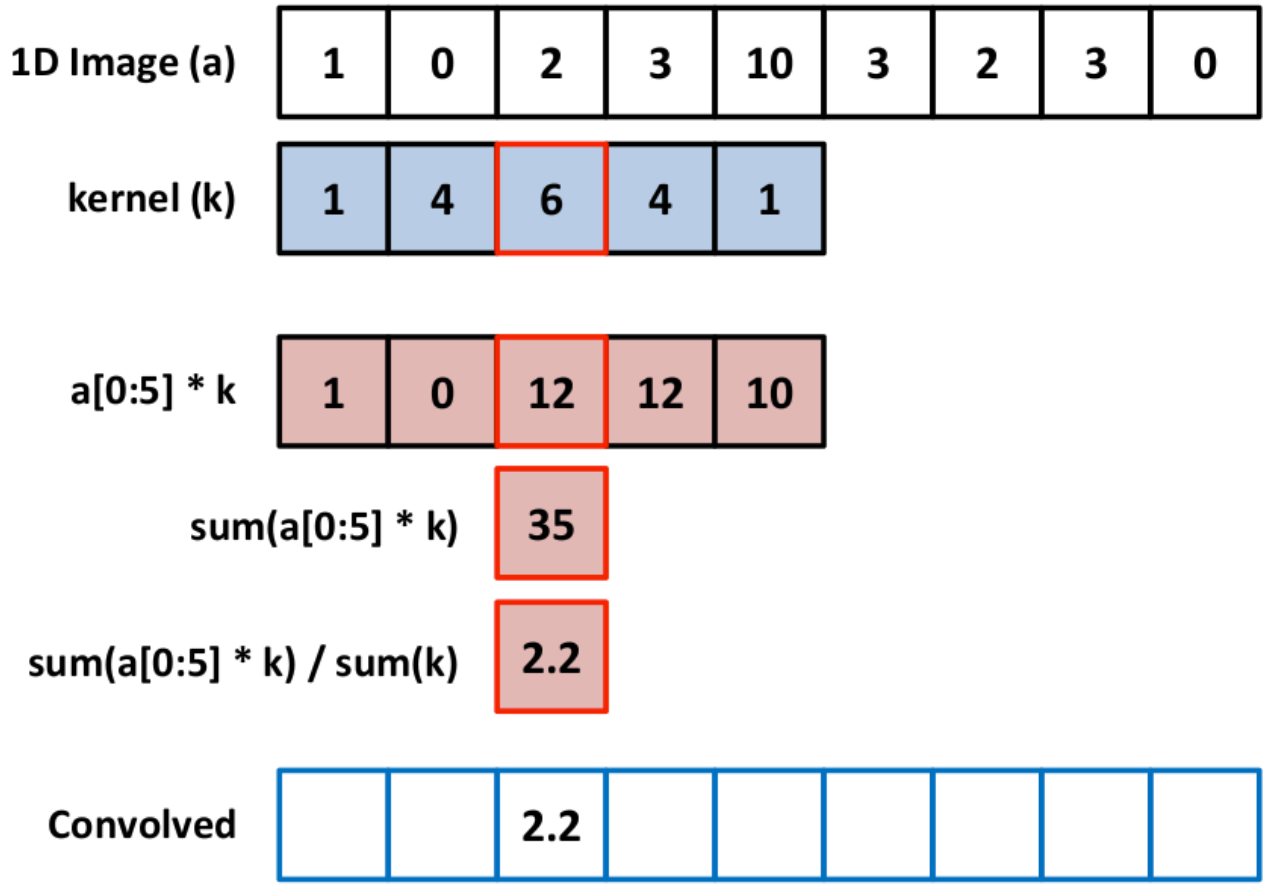
Digital image processing: Contrast adjustment



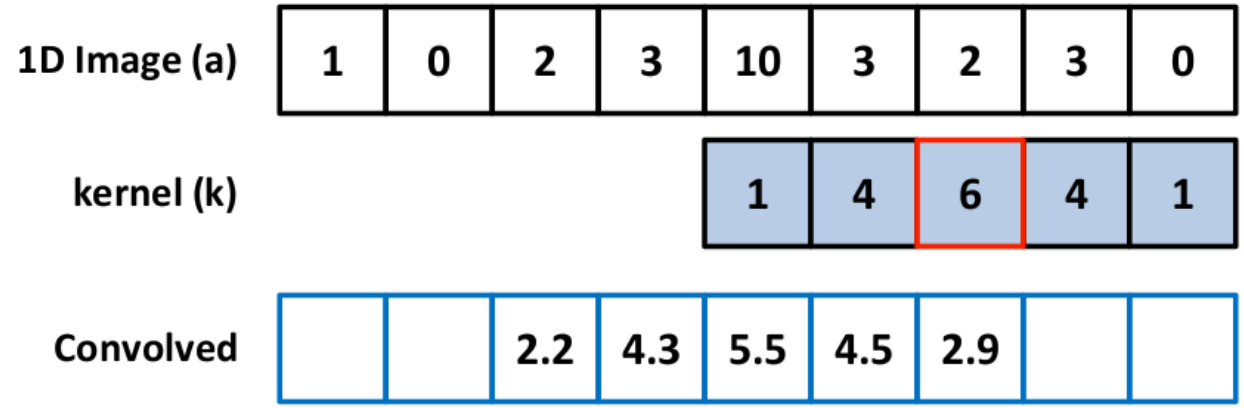
Digital image processing: Convolutions

- A convolution is a mathematical operation that produces a new version of an existing function using selections of that function and a second function
- In digital image processing this second function is called a kernel
- With the right kernel, convolutions can perform tasks and extract important features
  - Blurring – Gaussian kernel
  - Edge finding
  - Corner finding

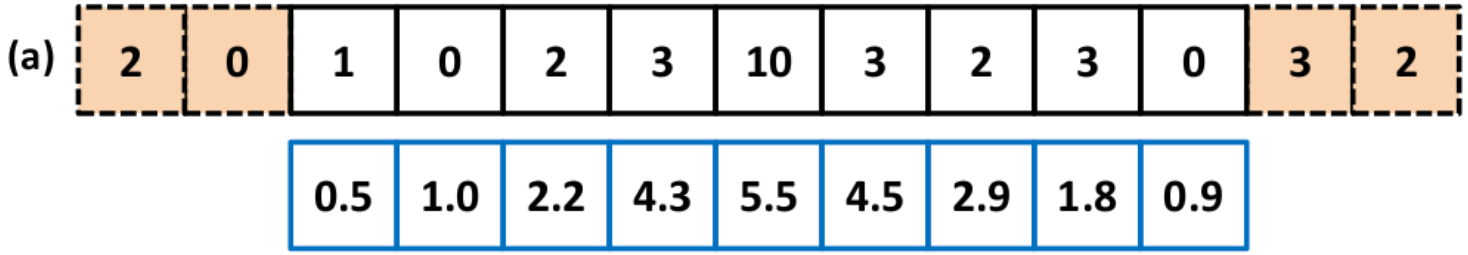
Digital image processing: Convolutions



Digital image processing: Convolutions

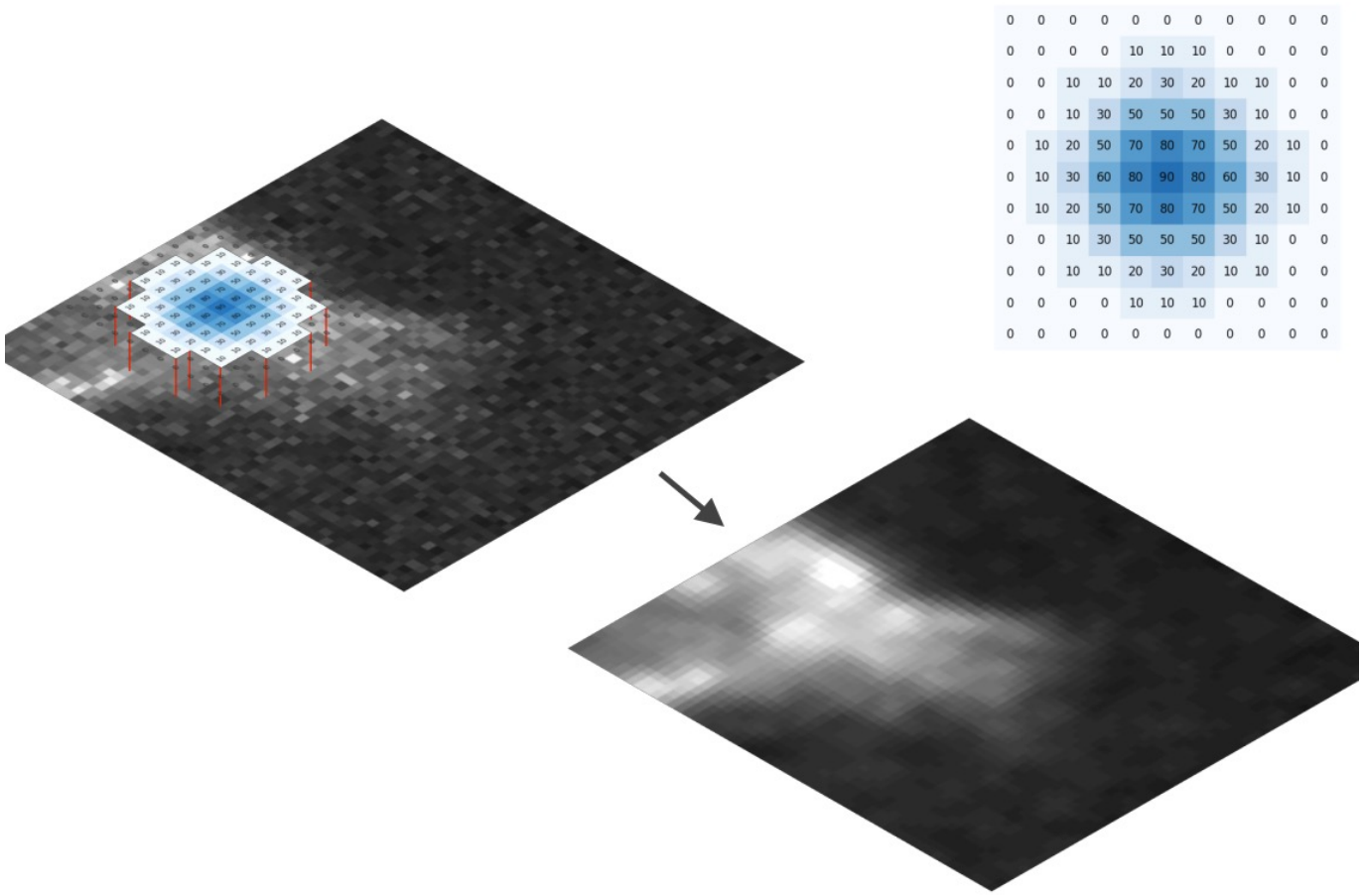


Note: Behavior at edges is undefined. Default in *scipy* is `reflect`.



Jonas Hartmann, Computational Image Analysis, 2017

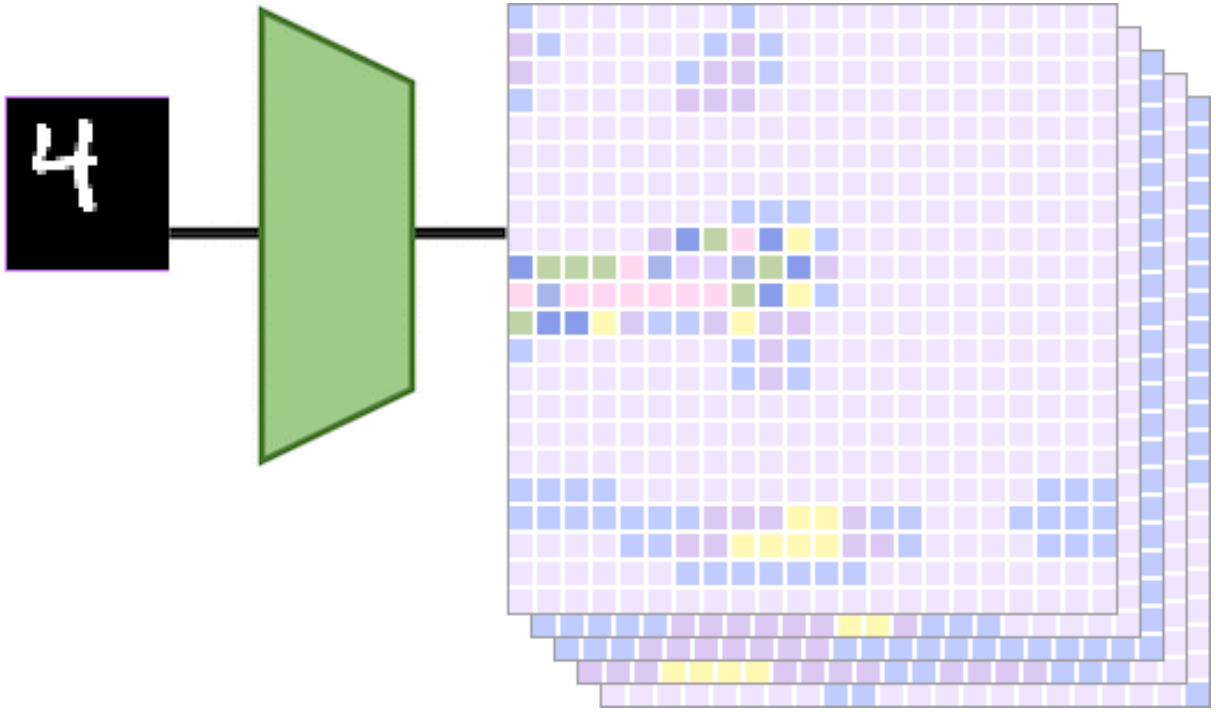
Digital image processing: Convolutions



Jonas Hartmann, Computational Image Analysis, 2017

Digital image processing: Convolutions

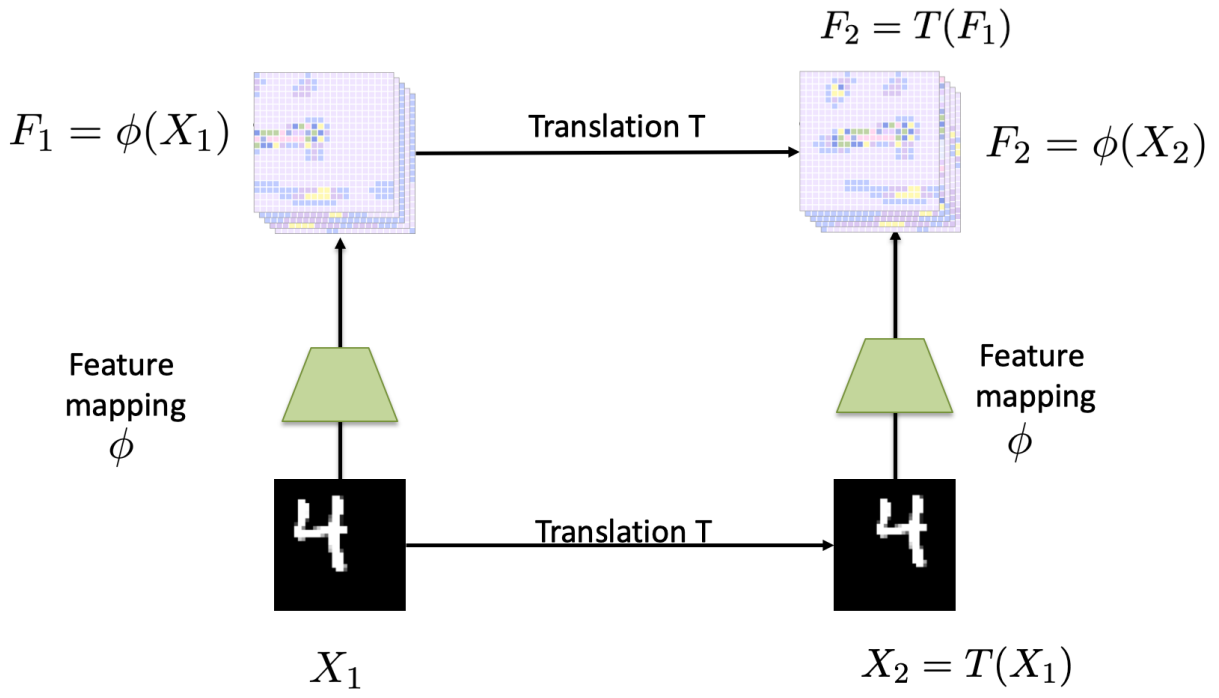
- Convolutions are a mathematical operation that “respect” translation



https://chriswolfvision.medium.com/what-is-translation-equivariance-and-why-do-we-use-convolutions-to-get-it-6f18139d4c59

Digital image processing: Convolutions

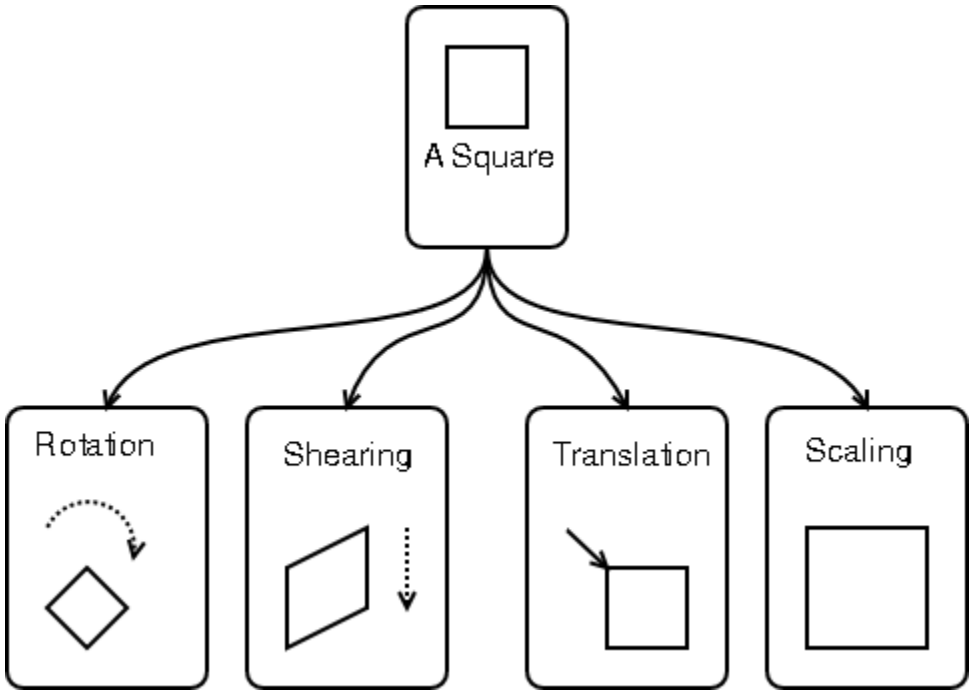
- The mathematical term for this "respect" is translational equivariance
- Equivariant mappings preserve the algebraic structure of transformations
  - The mapping commutes with the symmetry operation



https://chriswolfvision.medium.com/what-is-translation-equivariance-and-why-do-we-use-convolutions-to-get-it-6f18139d4c59

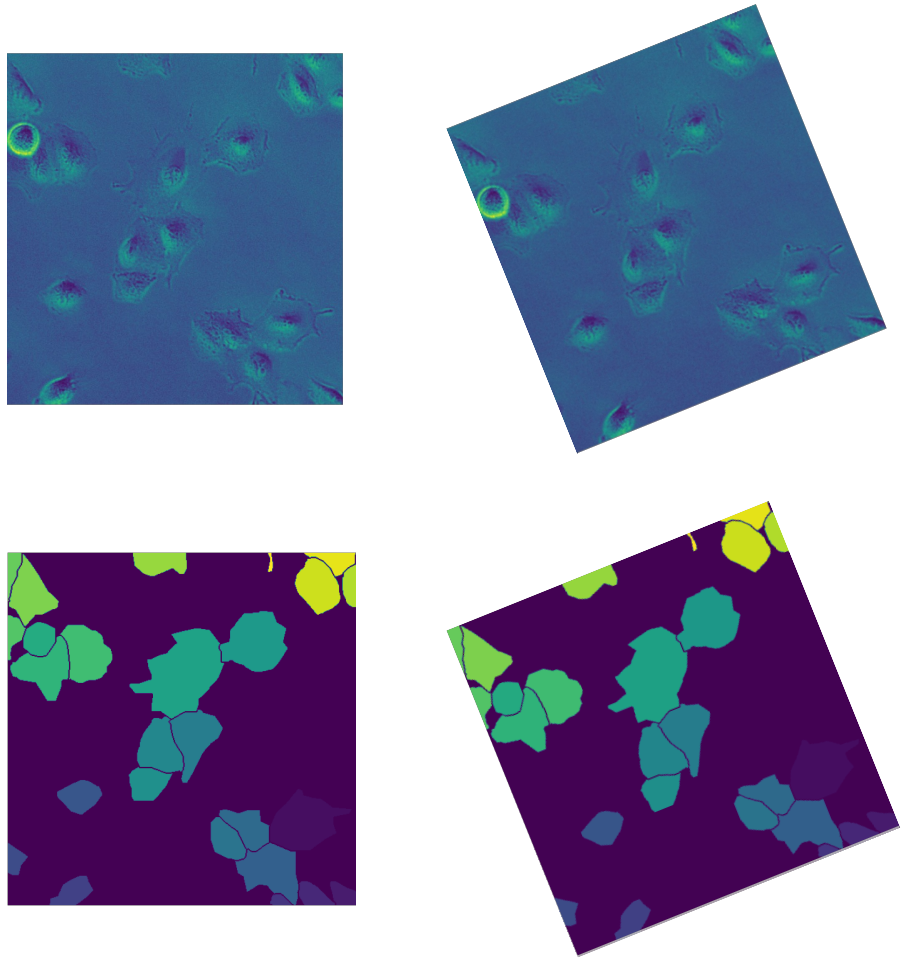
Digital image processing: Affine transforms

- Affine transformations transform a shape while preserving
  - Straight lines
  - Relative distances of points on straight lines
- We use affine transformations for image augmentation
- Image augmentation boosts the diversity of our training data by performing augmentation operations that preserve the labels

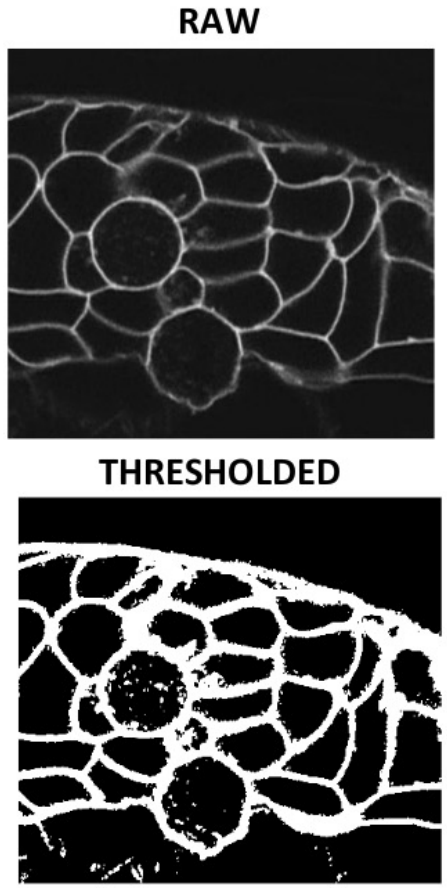


Digital image processing: Affine transforms

- When used for augmentation with pixel-level predictions, affine transformations need to be applied to both the image and the labels
  - Applying the transformation to just one corrupts the learning signal
- Affine transformations typically involve interpolation of some kind
  - Pay attention to how this happens!
  - Image precision also matters!



Digital image processing: Foreground detection



- Foreground detection splits the background and foreground of the image
  - The end result is a binary True/False array
  - This is also called a mask
- Thresholding is an essential part of foreground detection
  - Manual vs. automated
  - Uniform vs. adaptive
- Morphological operations can improve the quality of the final mask

Digital image processing: Thresholding

- There are many thresholding methods: Otsu's method is one that you should be aware of
- Otsu's method
  - Assume the image contains 2 classes of pixels following a bi-modal histogram
  - Search for the threshold that minimizes the intra-class variance
    - Minimizing the intra-class variance is the same as maximizing the inter-class variance

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

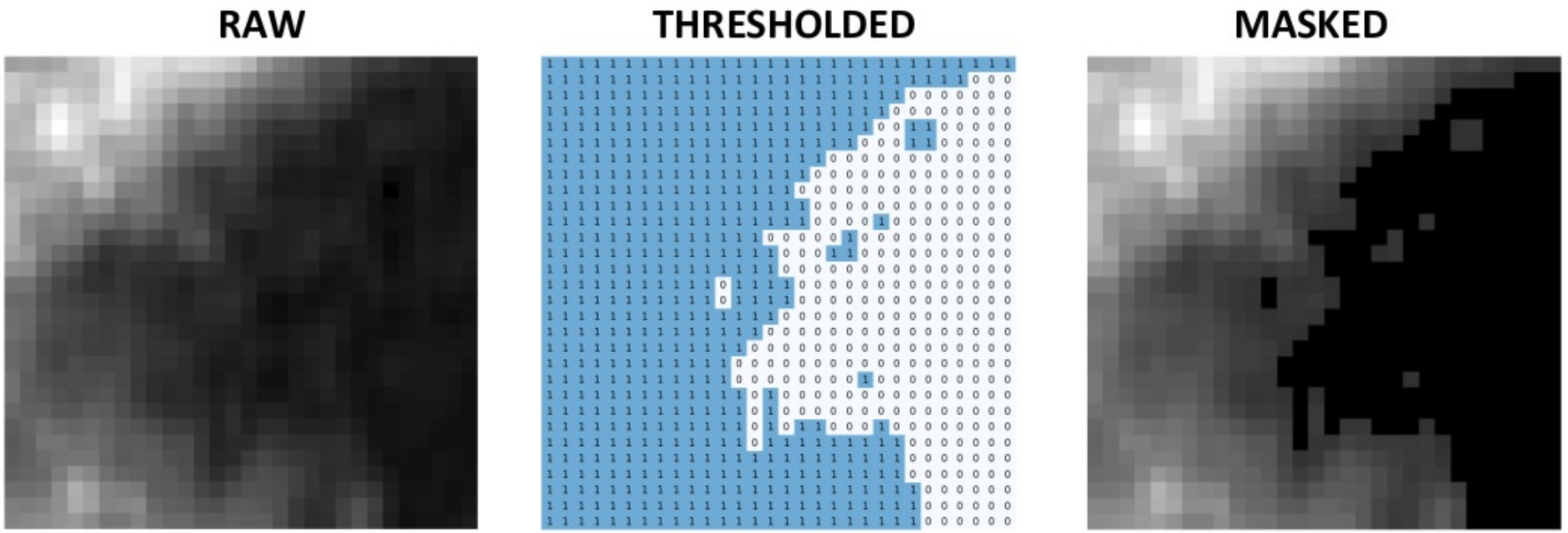
$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T$$
$$\omega_0 + \omega_1 = 1$$

Algorithm

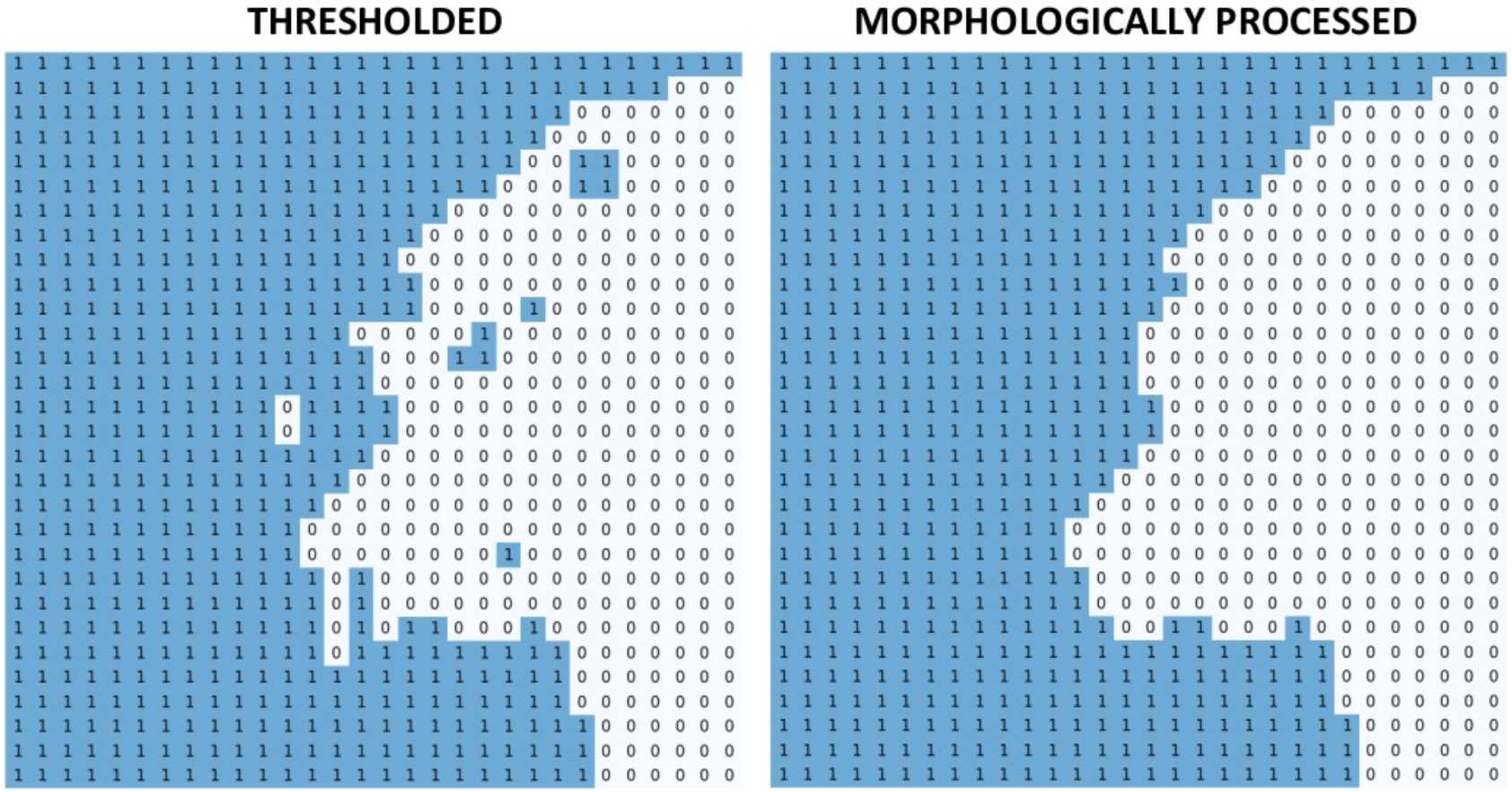
1. Compute histogram and probabilities of each intensity level
2. Set up initial $\omega_i(0)$ and $\mu_i(0)$
3. Step through all possible thresholds $t = 1, \ldots$ maximum intensity
    1. Update ω_i and μ_i
    2. Compute $\sigma_b^2(t)$
4. Desired threshold corresponds to the maximum $\sigma_b^2(t)$

Digital image processing: Thresholding

- Masks from thresholded images can be used to operate on other arrays/images

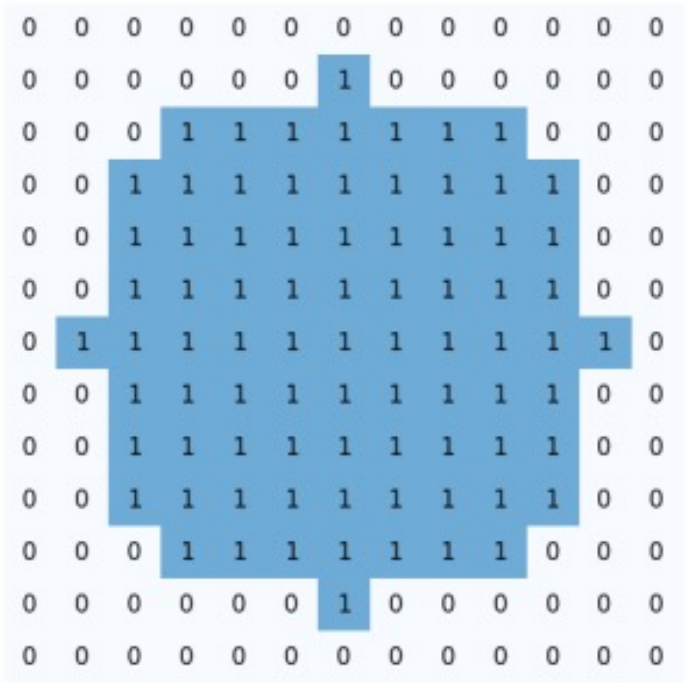


Digital image processing: Morphological operations



Digital image processing: Morphological operations

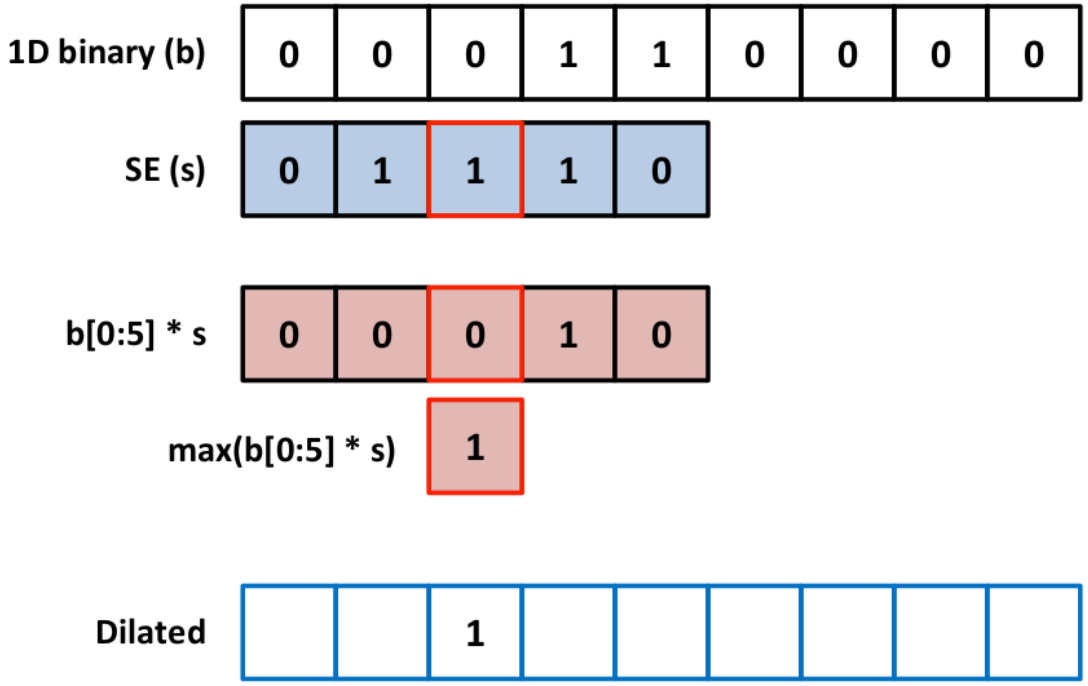
- Very similar in principle to filtering
  - “Structuring elements” are used in a similar manner as kernels
- Common operations
  - Erosion and dilation
  - Opening and closing
  - Hole filling

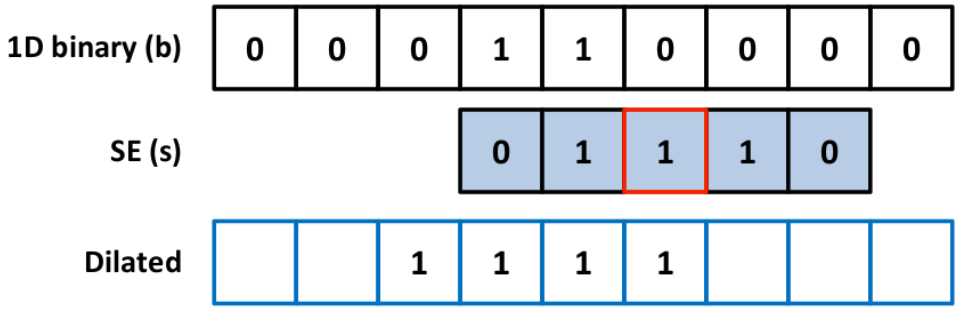


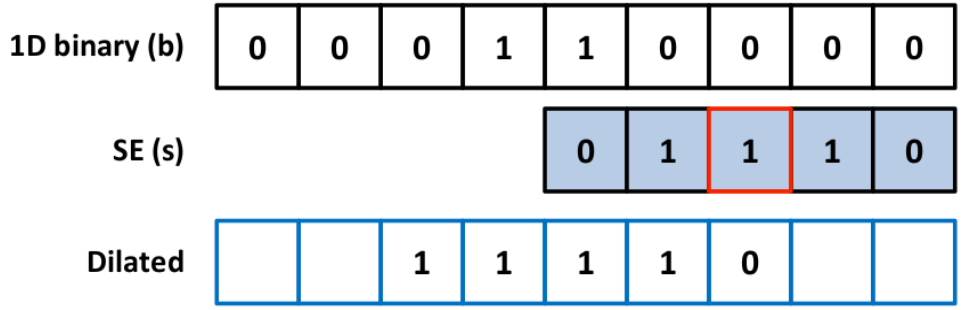
DISC-SHAPED SE

Digital image processing: Morphological operations

- Example: Dilation in 1D





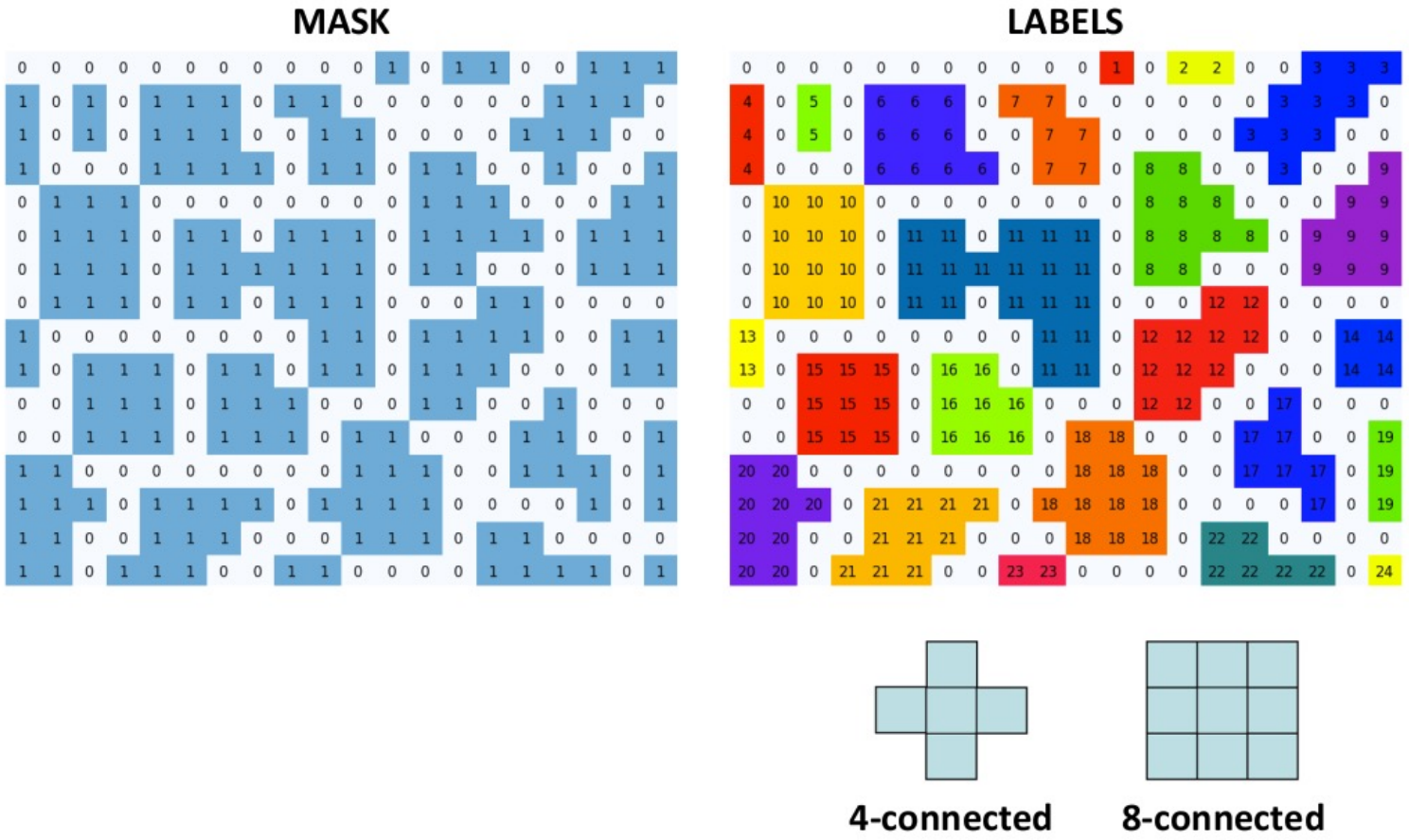


Jonas Hartmann, Computational Image Analysis, 2017

Digital image processing: Morphological operations

- Multiple operations and fine tuning may be required to get the desired result
  - This fine-tuning often introduces “brittleness” into the pipeline – it works well for data it was finetuned on and poorly for other sets
  - Bypassing the need for this kind of fine-tuning is a big advantage of deep learning
  - Classical methods like morphological operations can still have a role in building deep learning-enabled systems

Digital image processing: Connected components

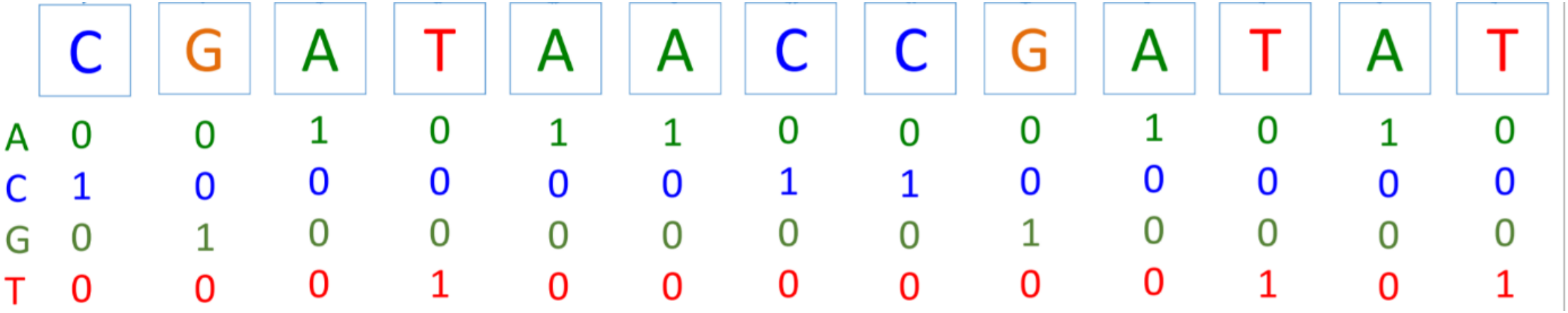


Jonas Hartmann, Computational Image Analysis, 2017

Sequences as data

- Typical examples: amino acid sequences (proteins), nucleotide sequences
  - Nucleotides: A, C, (T/U), G
  - Amino acids: A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V
  - Time series: EKG, EEG, etc.
- Sequences often need a numerical encoding to make them amenable to machine learning
  - Time series sequences are an exception
  - Even with a numerical encoding there are additional issues to make sequences ML ready (e.g., uneven length, padding, etc.)

One hot encoding



	C	G	A	T	A	A	C	C	G	A	T	A	T
A	0	0	1	0	1	1	0	0	0	1	0	1	0
C	1	0	0	0	0	0	1	1	0	0	0	0	0
G	0	1	0	0	0	0	0	0	1	0	0	0	0
T	0	0	0	1	0	0	0	0	0	0	1	0	1

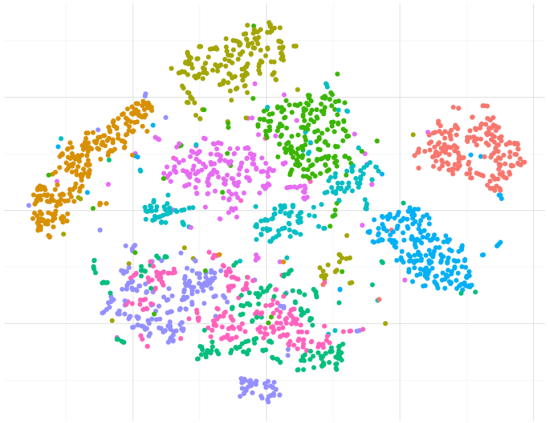
- Each token of a vocabulary gets labeled with all 0's and a single 1
- Pros:
  - Simple, fast, inexpensive

One hot encoding

	C	G	A	T	A	A	C	C	G	A	T	A	T
A	0	0	1	0	1	1	0	0	0	1	0	1	0
C	1	0	0	0	0	0	1	1	0	0	0	0	0
G	0	1	0	0	0	0	0	0	1	0	0	0	0
T	0	0	0	1	0	0	0	0	0	0	1	0	1

- Cons:
  - Sparse, memory inefficient, and high dimensional
  - Low information content – no notion of similarity between sequence or structural elements
  - Dimensionality is fixed

Learned embeddings



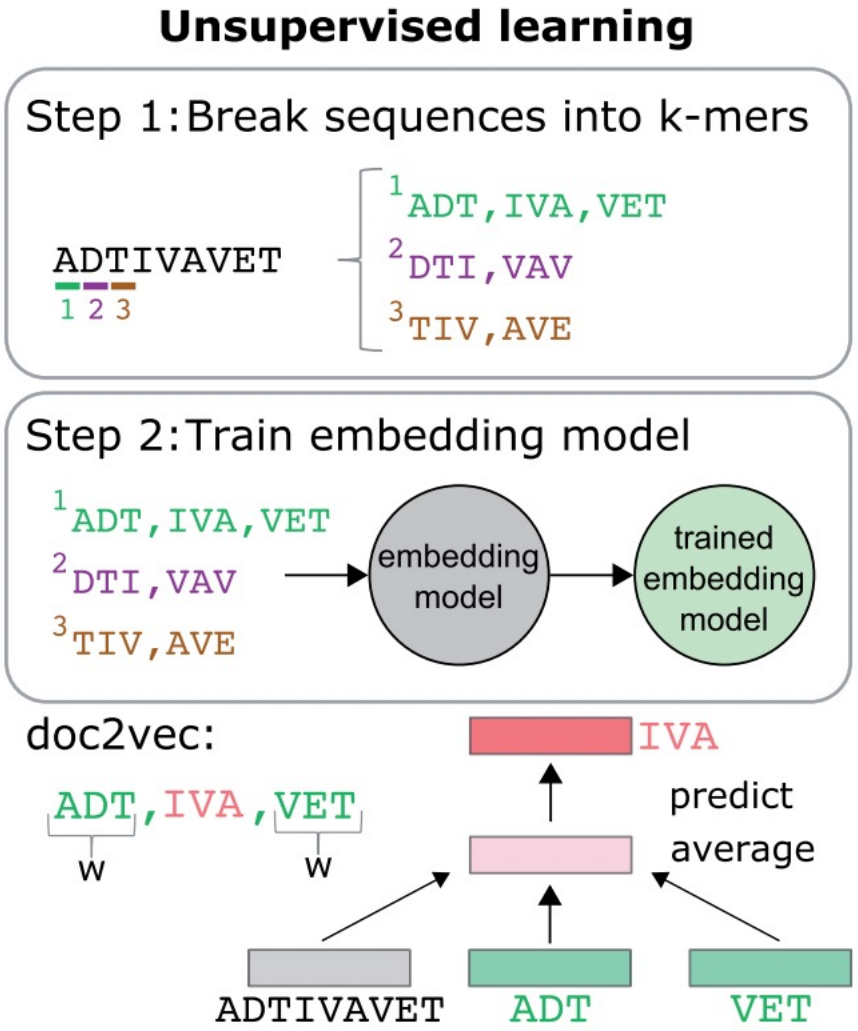
Map a token to a vector and require this mapping to have certain properties

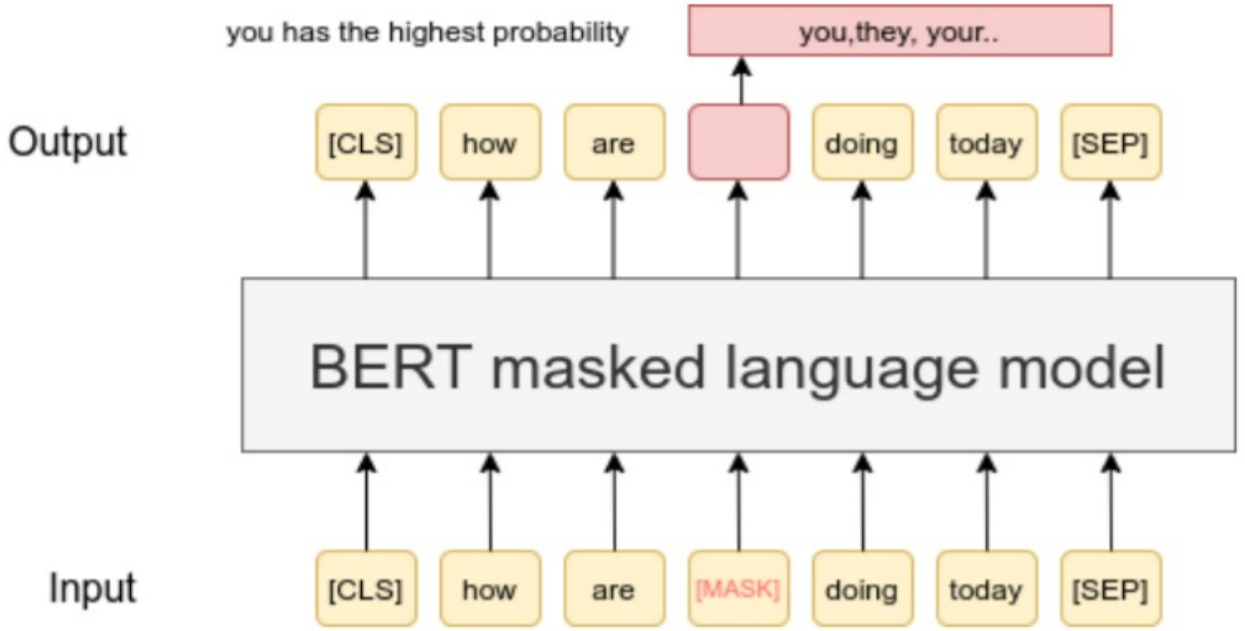
$$X \rightarrow \vec{k}; k \in R^n$$

- Learned embeddings seek to map tokens to a vector space in a particular way to ensure that the embedding has specific properties
  - Similar tokens are close together, while dissimilar tokens are far apart
  - Embeddings have information about the sequence context
- There have been a variety of methods developed to create useful embeddings
  - word2vec, doc2vec
  - Unsupervised k-mer sequence embeddings
  - ESM Fold, AlphaFold, etc.

https://doi.org/10.1093/bioinformatics/bty178
https://www.pnas.org/doi/full/10.1073/pnas.2016239118
https://www.nature.com/articles/s41586-021-03819-2

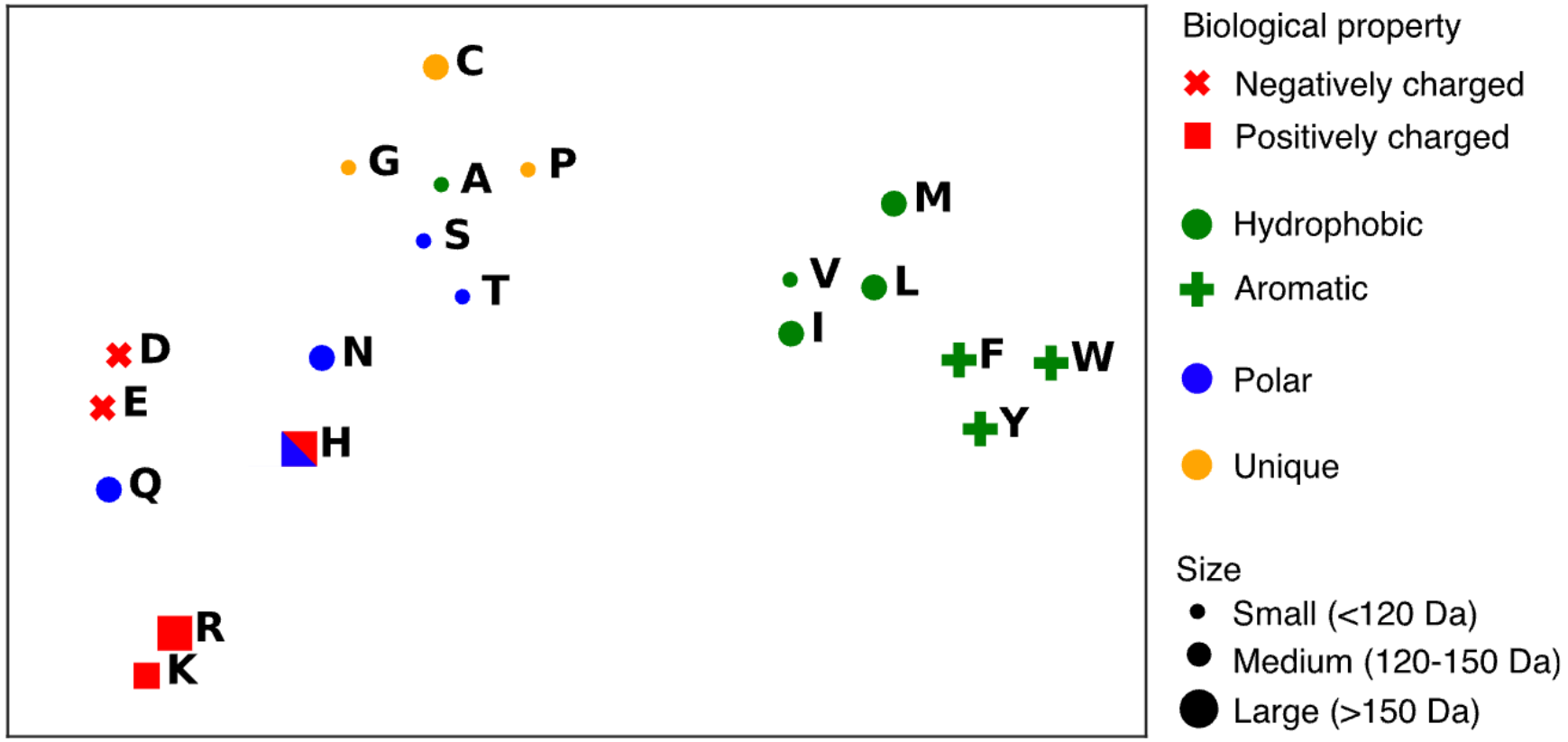
Learned embeddings





Masked pre-training is a common technique with language models and has been successful on protein sequences

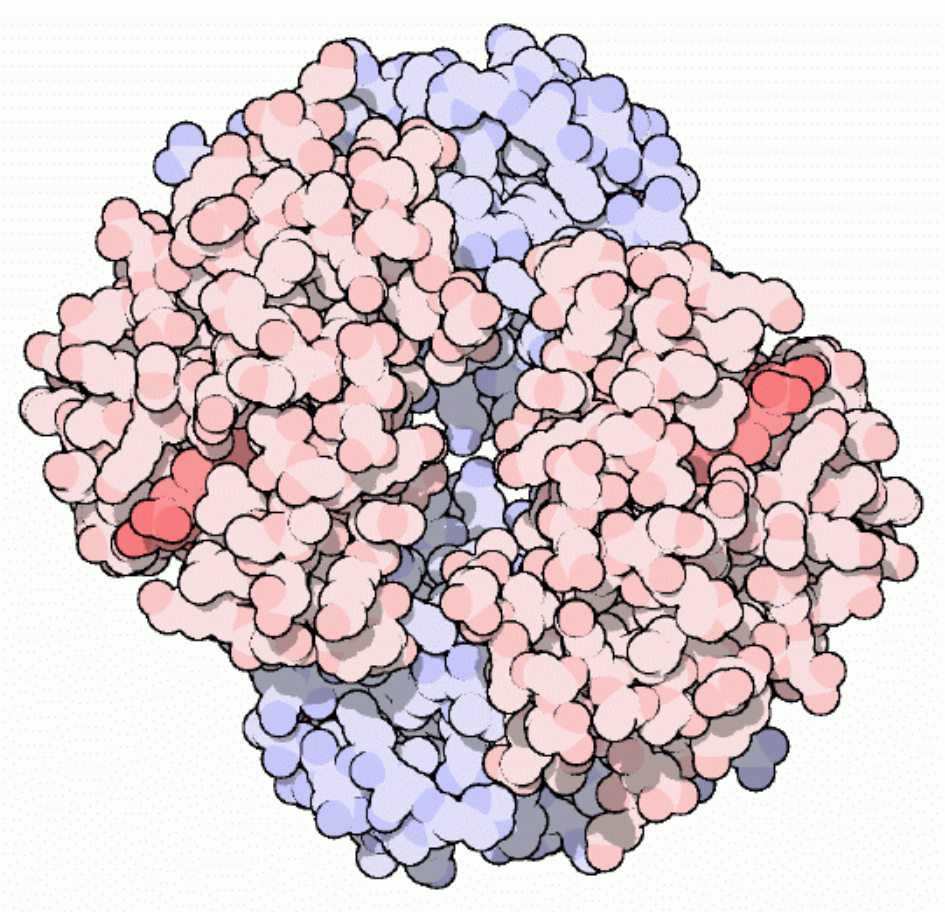
Learned embeddings



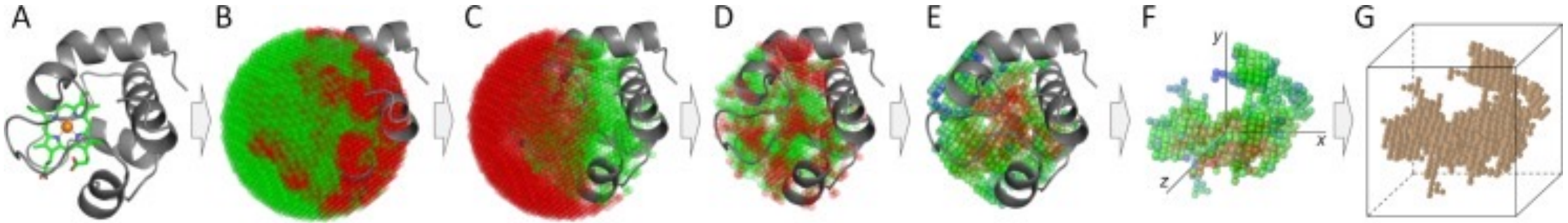
Embeddings learned by ESM clustered amino acids by their side chain properties

Structures as data

- Structural data captures the spatial locations of the component atoms in a molecule
- This data can be represented in a variety of ways so it can be presented to deep learning models
- Each representation lends itself to a different set of deep learning methods



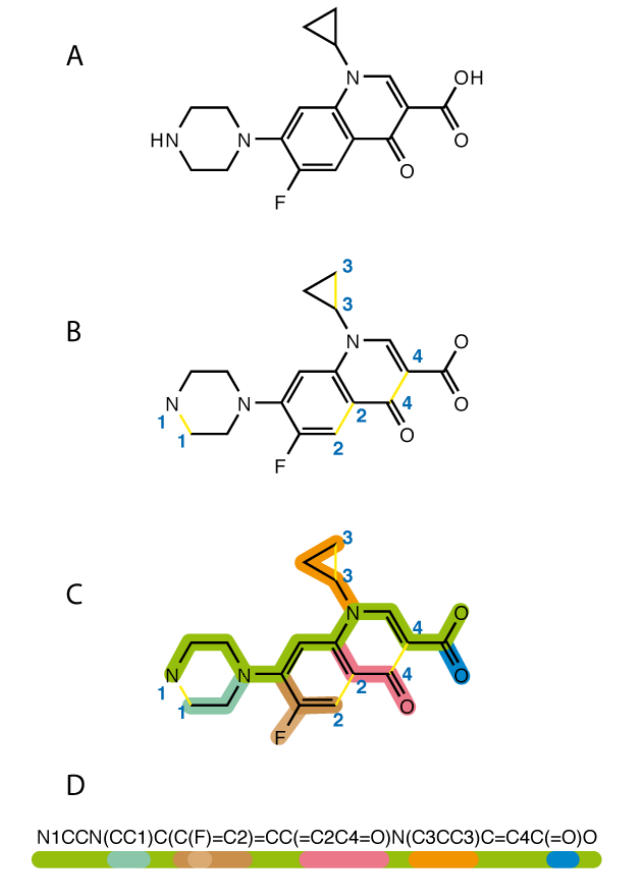
Structures as data



- Voxel representation: Structural information is captured by assigning features to a 3D grid
  - Features can be occupancy, chemical properties, etc.
  - Voxel representations effectively convert structures to 3D images, which makes them amenable to vision models

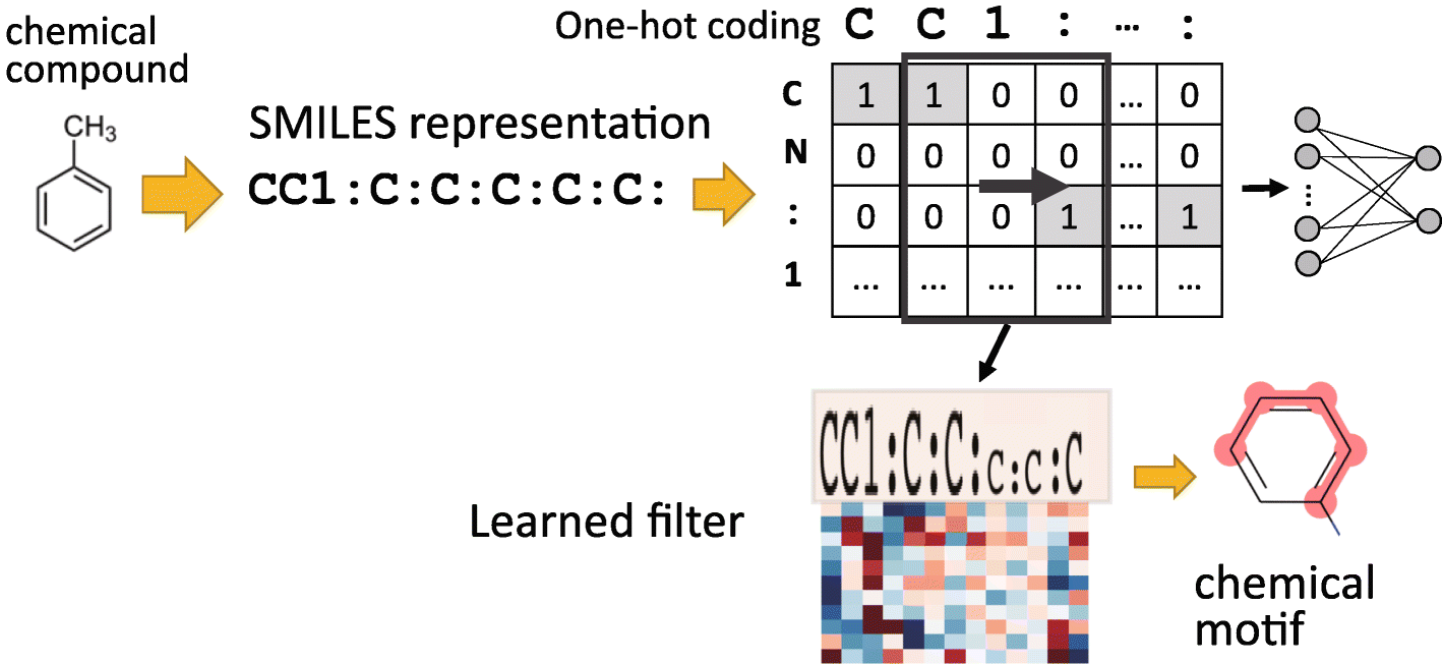
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6375647/

Structures as data



- SMILES
  - Simplified molecular input line entry system
  - ASCII representation for chemical structures
  - Typically used for small molecules
  - Amenable to both language models and vision models
  - One-hot encoding on top of SMILES representations has seen some use

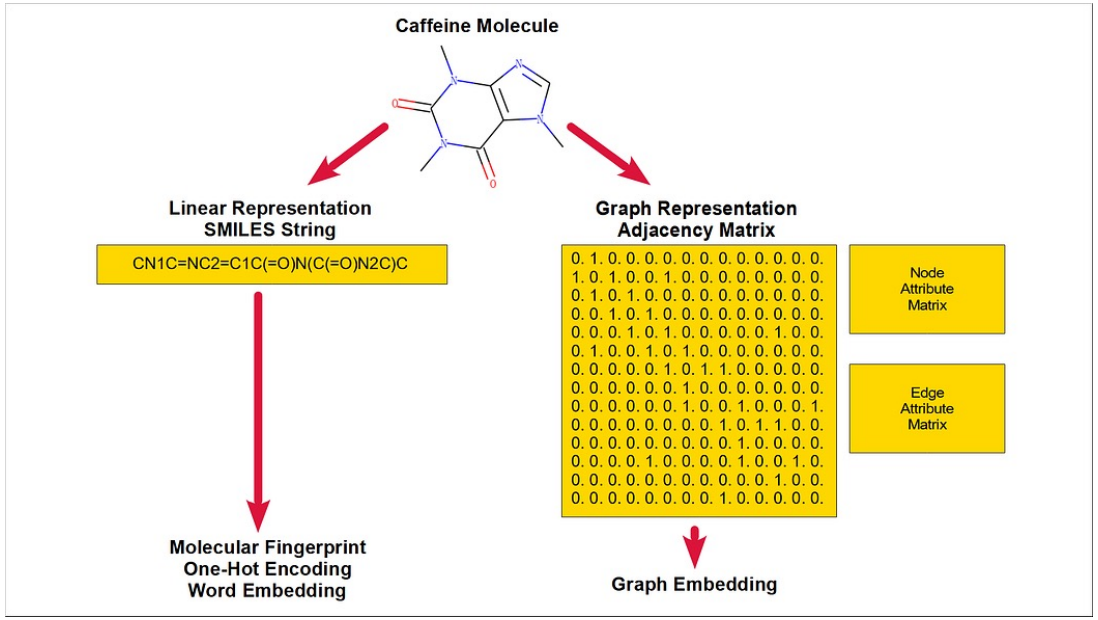
Structures as data



One-hot encoding on top of SMILES representations has seen some use

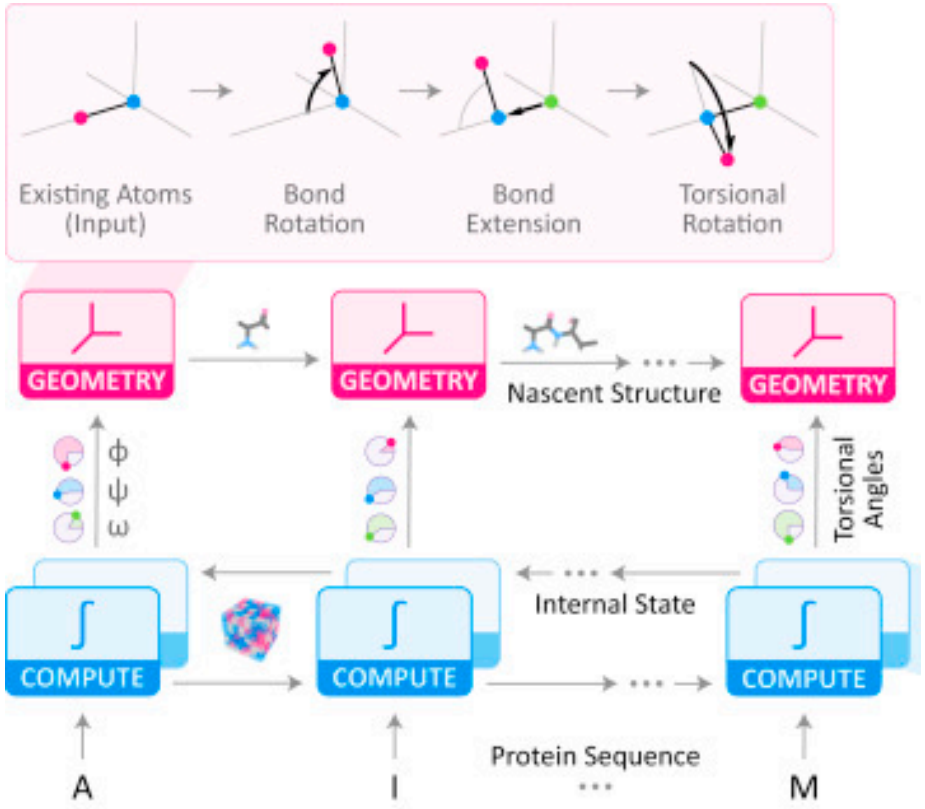
https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2523-5

Structures as data



- Graph representation
  - Atoms are nodes
  - Bonds are edges
  - The graph can have node and edge features that are captured by their respective matrices
  - The adjacency matrix captures the atom connectivity
  - The graph representation is amenable to graph neural networks

Structures as data



- Molecular point cloud
  - Version of the graph representation where the 3D coordinates of atoms are captured
  - Different variants of this idea exist
    - Coordinates as node features
    - Encoding location via torsion angles
  - Different levels of resolution also exist
    - e.g., only capture the location of the Cα carbon for amino acids
  - Amenable to graph neural networks

https://www.nature.com/articles/s41587-022-01432-w