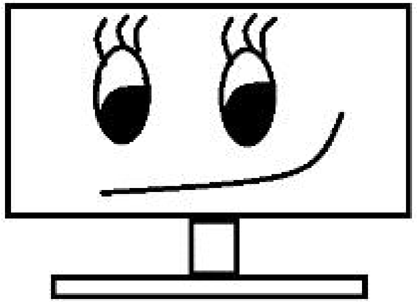
Seneca



CVI620/ DPS920
Introduction to Computer Vision

Image Segmentation and Deep Learning

Seneca College

Vida Movahedi

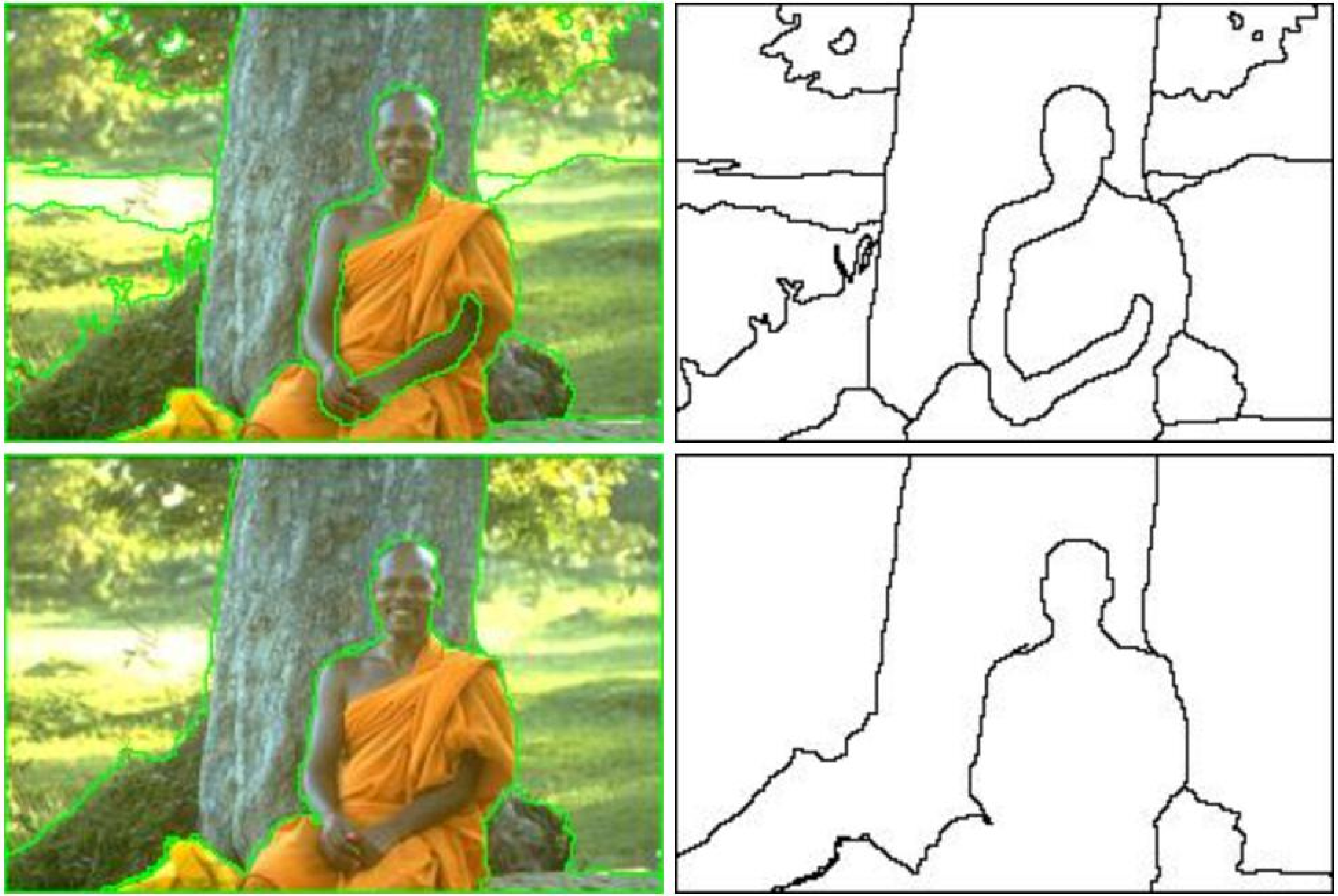
Overview

- Image vs. Object Segmentation
- Methods
  - Active Contour Methods
  - Perceptual Grouping or Contour Grouping Methods
  - Regional Methods
    - Region Growing
    - Grab-cut (graph-based)
    - Mean-shift
  - Deep Learning methods
- Introduction to Deep Learning

Image Segmentation

- Segmentation is a way to separate the image into regions with **homogenous behavior**
- Image segmentation is the task of finding groups of pixels that "go together" [1]
- Partition an image into regions, each corresponding to an **object** or **entity** [4]
- Each pixel is assigned a label such that pixels with the same label share certain properties, e.g. belonging to the same object (or background) [4]
- Many methods! Just mentioning a very few here!

- Various levels of detail
- Different when done by multiple human subjects



[Source: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/]

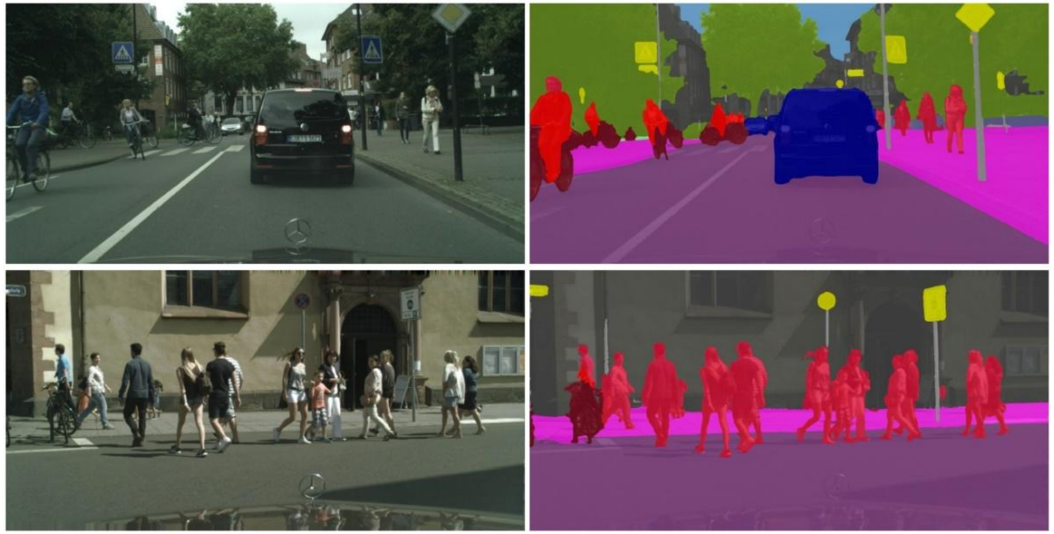
Object Segmentation

- Image segmentation is not the same as object segmentation.
- In salient **object segmentation**, the goal is to group pixels that belong to the **same object**;
- In other words, find the boundary between **foreground** (object) and **background**.

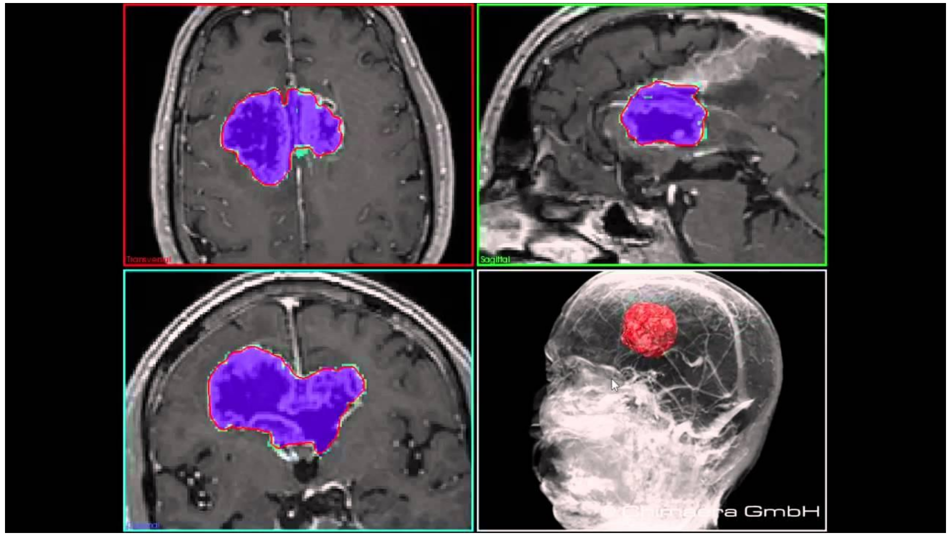
Segmentation: Two Perspectives! [4]

Figure 1.1: Salient object segmentation - (a) A sample image (source: [3]), (b) Segmentation as regional labeling, (c) corresponding object boundary.

How Can We Solve the Image Segmentation Problem?



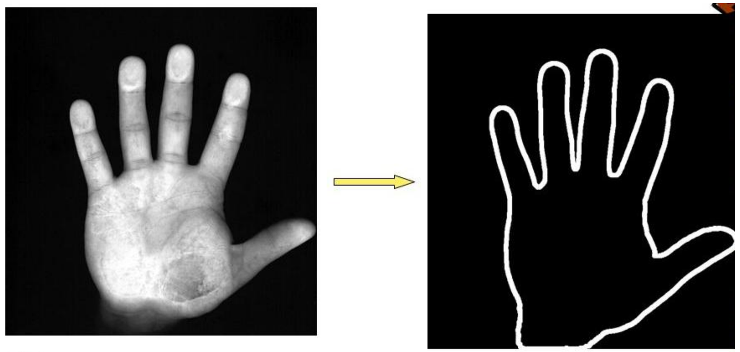
http://vladlen.info/wp-content/uploads/FSO-1.jpg



https://i.ytimg.com/vi/7wCC2NaVLjs/maxresdefault.jpg

https://vision.in.tum.de/_media/spezial/bib/hazirbas2014msc.jpg

https://vision.in.tum.de/_media/spezial/bib/nieuwenhuis-cremers-pami12_2.jpg



http://slideplayer.com/slide/4592950/15/images/16/Image+segmentation.jpg

Active Contour Methods

Active Contour Methods

- Also called **snakes**
- The active contour model is a method to **fit** open or closed splines **to lines or edges** in an image [5].
- It works by **minimizing an energy** that is in part defined by the image and part by the spline’s shape: length and smoothness. The minimization is done implicitly in the **shape energy** and explicitly in the **image energy**. [6]

- [5] Snakes: Active contour models. Kass, M.; Witkin, A.; Terzopoulos, D. International Journal of Computer Vision 1 (4): 321 (1988). DOI:10.1007/BF00133570
- [6] http://scikit-image.org/docs/dev/auto_examples/edges/plot_active_contours.html

Snakes and Energy

- A simple **elastic** snake is defined as a curve

$$f(s) = \big(x(s),\, y(s)\big),\ s \in [0,1]$$

- Goal: minimize the energy of the snake

$$E_{\text{snake}} = E_{\text{internal}} + E_{\text{external}} = E_{\text{internal}} + E_{\text{image}} + E_{\text{con}}$$

$E_{internal}$:	Controls the deformation of the snake. A smooth and continuous snake has a lower internal energy than a stretched snake or one with high curvature (the elasticity and smoothness)
E_{image}:	Controls the fitting of the contour to the image. A snake which passes through dark ridges, edges with strong gradient (high magnitude), or corners and line terminations has a lower image energy than one that does not pass through these (how well the contour contracts or expands to the edges)
E_{con}:	User constraints or prior model constraints. A snake that is close (or away) from user specified features has a lower constraint energy than one that does not

https://www.youtube.com/watch?v=3-YQ9k47ktY

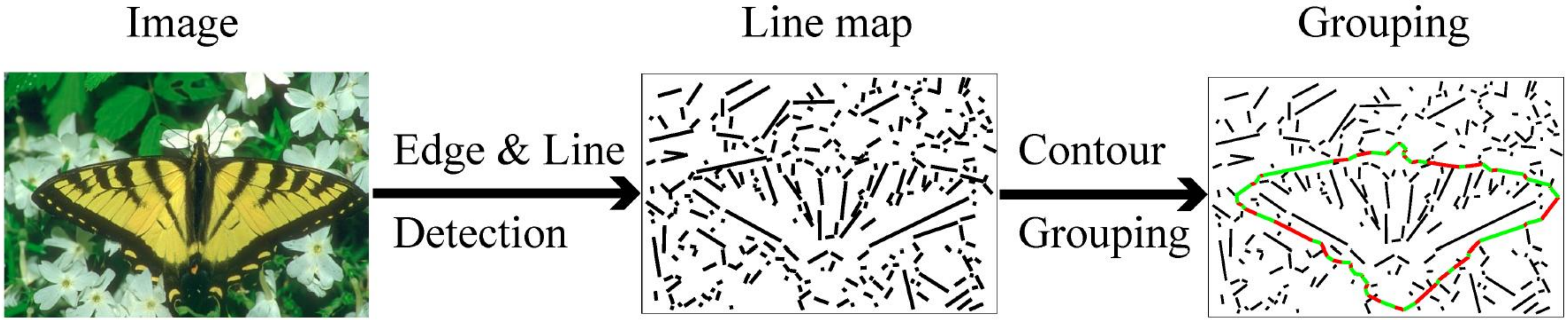
Region Growing

Grab-Cut (Graph-Based)

Mean-Shift

Contour Grouping Methods

- Grouping edges or line approximations of edges into a closed boundary of an object [4]



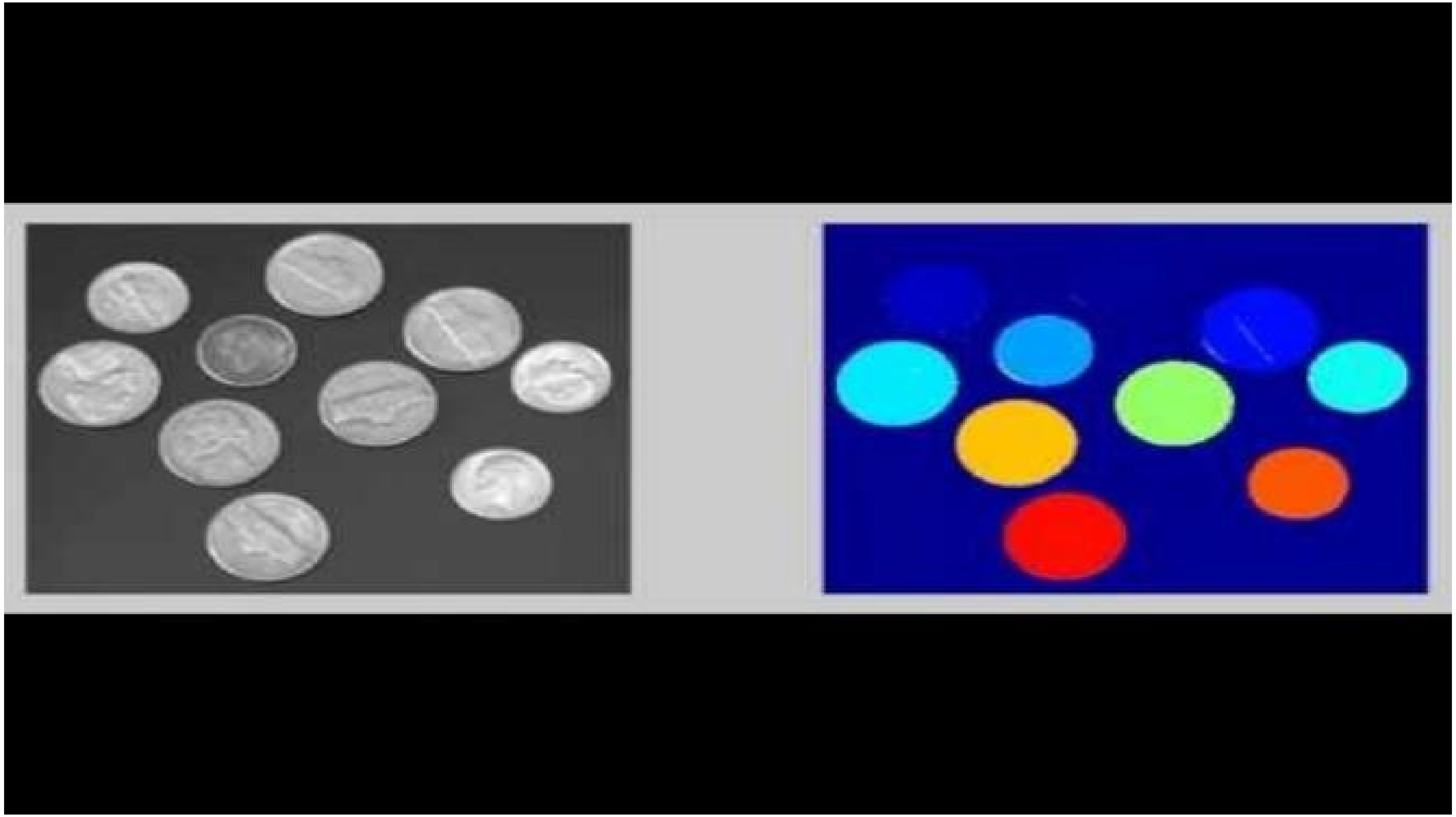
Region Growing Methods [2]

- **Flood Fill**
  - A seed point is selected from an image
  - All ***similar*** neighboring pixels are labelled with the **same label** (belonging to the same region)
  - **Similarity** is defined as being within a range of the **seed** or **neighbors**

Procedure

- A seed point is selected, then all **similar** neighboring points are colored with a uniform color
- Neighboring pixels need not be identical in color, but within a specified **range** (loDiff to upDiff) of either the current pixel or the original seed value

Region Growing Methods



https://www.youtube.com/watch?v=SokLiR6PA0I

OpenCV Flood Fill [2]

```
int cv::floodFill(
  cv::InputOutputArray image,                    // Input image, 1 or 3 channels
  cv::Point            seed,                     // Start point for flood
  cv::Scalar           newVal,                   // Value for painted pixels
  cv::Rect*            rect,                     // Output bounds painted domain
  cv::Scalar           lowDiff  = cv::Scalar(),// Maximum down color distance
  cv::Scalar           highDiff = cv::Scalar(),// Maximum up color distance
  int                  flags                     // Local/global, and mask-only
);
```

Python:

cv.**floodFill**(image, mask, seedPoint, newVal[, loDiff[, upDiff[, flags]]]) -> retval, image, mask, rect

mask: to control where filling is done

Region Growing Methods [2, 5]

Watershed

- It is useful if we do not have the benefit of separate background mask
- An image is interpreted as a height field or landscape. Uniform areas are **valleys**, lines are **mountains**
- Fill the valley with water, until water from two valleys are about to merge
- Build a barrier to prevent merging
- Resulting barrier is the image segmentation

Procedure

- Takes the gradient of intensity to form valleys and mountains
- Start flooding the landscape at all local minima
- Wherever different evolving components meet, label the ridges (barriers)

OpenCV Watershed [2]

- An implementation of the watershed algorithm
- Allows the user to mark parts of an object or background
- It tells the algorithm to "group points like these together"

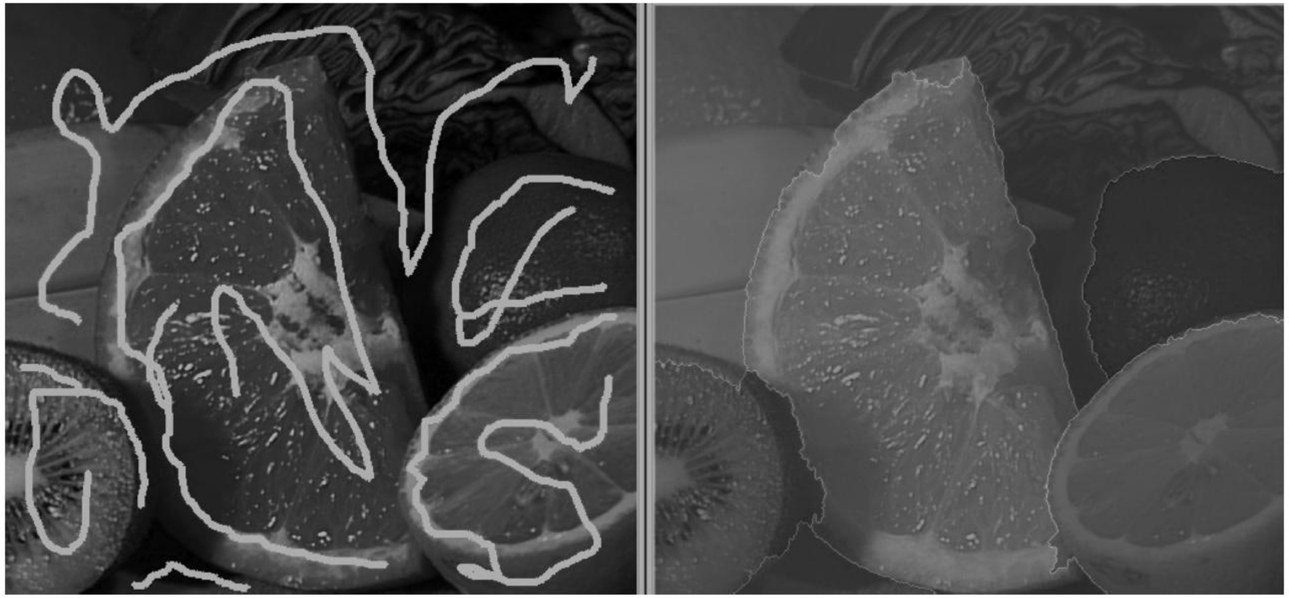


Figure 12-12. Watershed algorithm: after a user has marked objects that belong together (left), the algorithm merges the marked area into segments (right)

Watershed (cont.)

```
void cv::watershed(
  cv::InputArray       image,                  // Input 8-bit, three channels
  cv::InputOutputArray markers                 // 32-bit float, single channel
);
```

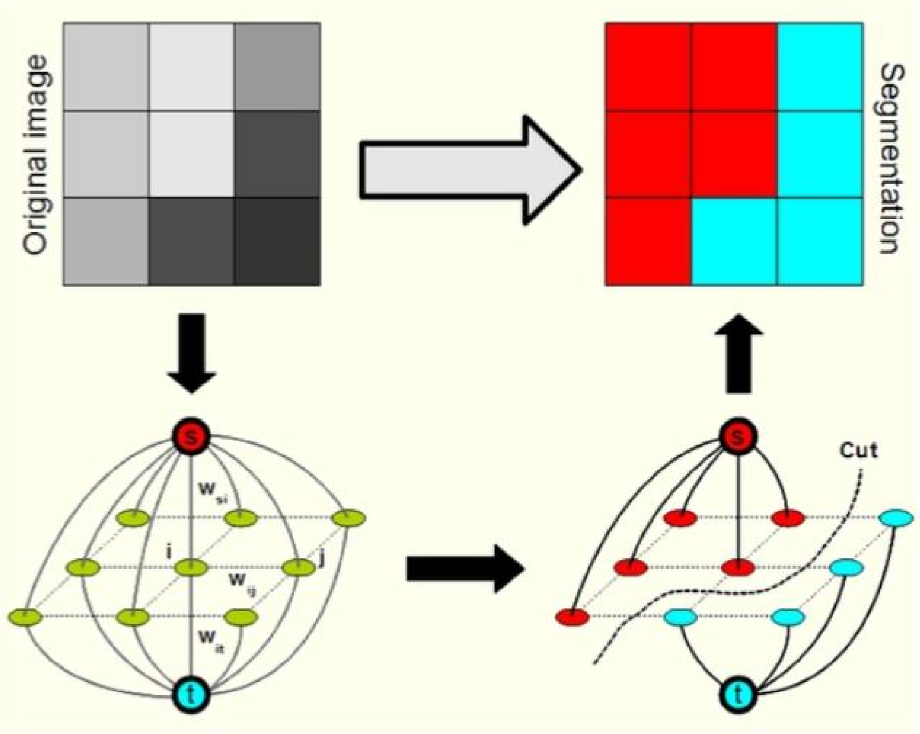
Python: cv.watershed(image, markers) -> markers

- The second argument, markers, is a single-channel integer (cv::S32) image, having the same dimension, and 0 everywhere except at caller's marks
- For example, the orange might have been marked with a "1," the lemon with a "2," the lime with "3," the upper background with "4," and so on
- The function will set all 0 pixels of markers to one of the marked values, except the boundaries between regions, which may be set to -1.

Region Growing
Grab-Cut (Graph-Based)
Mean-Shift

Grab-Cut [2, 5]

- A graph method: capable of obtaining excellent segmentations, often with just a bounding rectangle around the object
- It uses user-labeled foreground and background regions to establish distribution histograms
- Unlabeled foreground or background should conform to similar distributions (smooth and connected)
- Assertions are combined into an energy functional: a low energy to solutions that conforms to these assertions and a high energy to solutions that violates them.
- Final result is by minimizing this energy function



OpenCV Grab-Cut

```
void cv::grabCut(
  cv::InputArray       img,
  cv::InputOutputArray mask,
  cv::Rect             rect,
  cv::InputOutputArray bgdModel,
  cv::InputOutputArray fgdModel,
  int                  iterCount,
  int                  mode     = cv::GC_EVAL
);
```

Python:

```
cv.grabCut(  img, mask, rect, bgdModel, fgdModel,
iterCount[, mode]  ) -> mask, bgdModel, fgdModel
```

Region Growing
Grab-Cut (Graph-Based)
Mean-Shift

OpenCV Mean-Shift

- Mean-Shift looks at the **spatial** distribution of color (x,y)
- OpenCV implementation color, replace value with **mean**
- Segmentation is done over a scale pyramid
- Output is a "posterized" image (fine texture is removed and gradient is color are flattened)

```
Python:
cv.pyrMeanShiftFiltering
(src, sp, sr[, dst[,
maxLevel[, termcrit]]]
   ) -> dst
```

```
void cv::pyrMeanShiftFiltering(
  cv::InputArray    src,                          // 8-bit, Nc=3 image
  cv::OutputArray   dst,                          // 8-bit, Nc=3, same size as src
  cv::double        sp,                           // Spatial window radius
  cv::double        sr,                           // Color window radius
  int               maxLevel = 1,                 // Max pyramid level
  cv::TermCriteria termcrit = TermCriteria(
    cv::TermCriteria::MAX_ITER | cv::TermCriteria::EPS,
    5,
    1
  )
);
```

```
cv::pyrMeanShiftFiltering( src, dst, 20, 40, 2);
```

Figure 12-13. Mean-shift segmentation over scale using cv::pyrMeanShiftFiltering() with parameters max_level=2, spatialRadius=20, and colorRadius=40; similar areas now have similar values and so can be treated as super pixels (larger statistically similar areas), which can speed up subsequent processing significantly

New Methods: Deep Learning
Semantic Image Segmentation – Microsoft Ignite

https://youtu.be/FroRjEejA30

New Methods: Deep Learning
Segment Anything – Meta AI CV Research

- Segment Anything Model (SAM) is a new AI model from Meta AI that can "cut out" any object, in any image, with a single click
- The Project is a new task, model, and dataset for image segmentation
- https://segment-anything.com/
- https://www.youtube.com/shorts/oYUcl_cqKcs

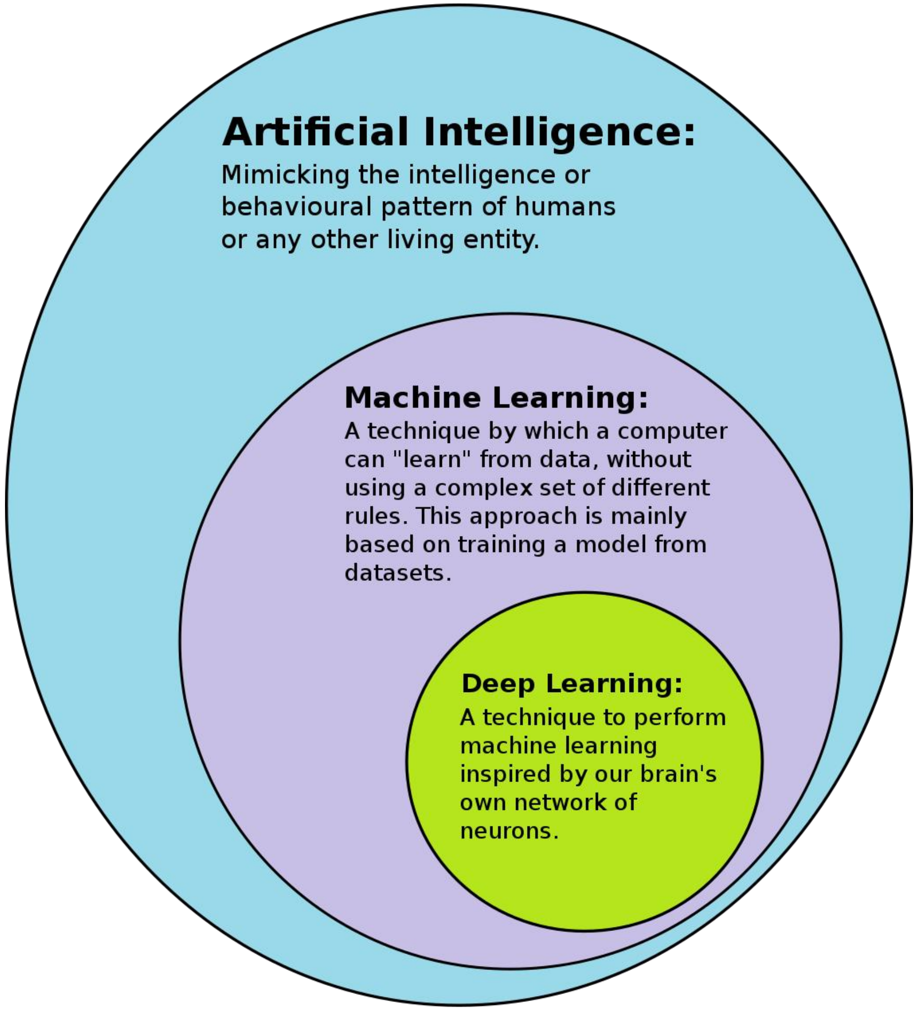
Quick Introduction to Deep Learning: DL-Based (Method 3)

Reminder

- Computer Vision problems can be solved using three approaches:
  1) Start from intuition and design a method, or
  2) Start from data and use Machine Learning, or
  3) Start from LOTS of data and use Deep Learning

Deep Learning

- A technique for implementing Machine Learning
- Allows for learning models
- Learning algorithm that is inspired by how biological brains work

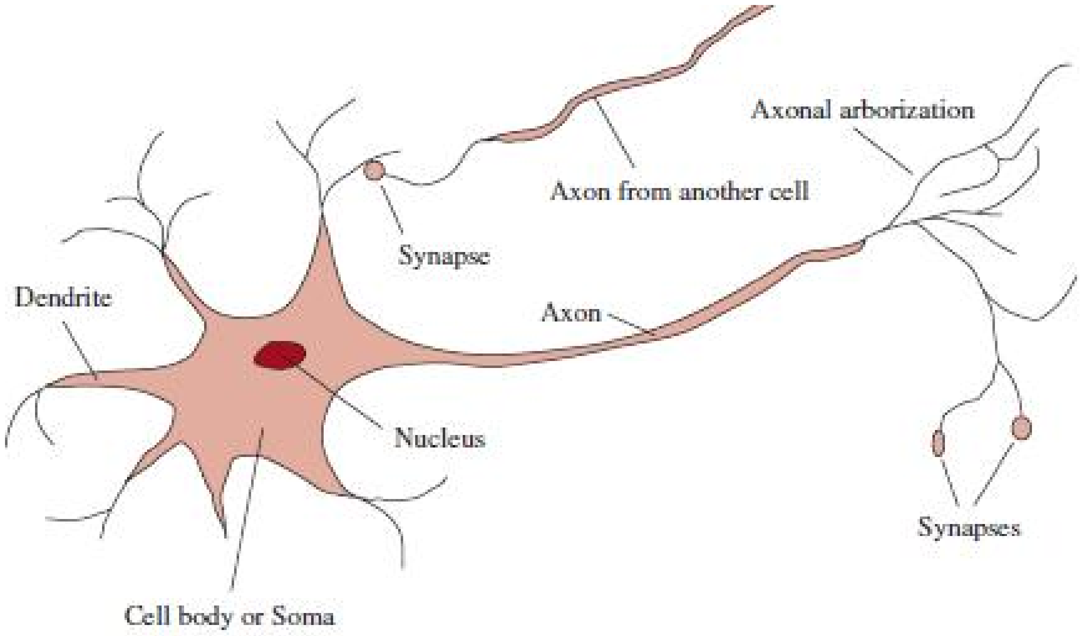


[https://en.wikipedia.org/wiki/Deep_learning]

Neuron

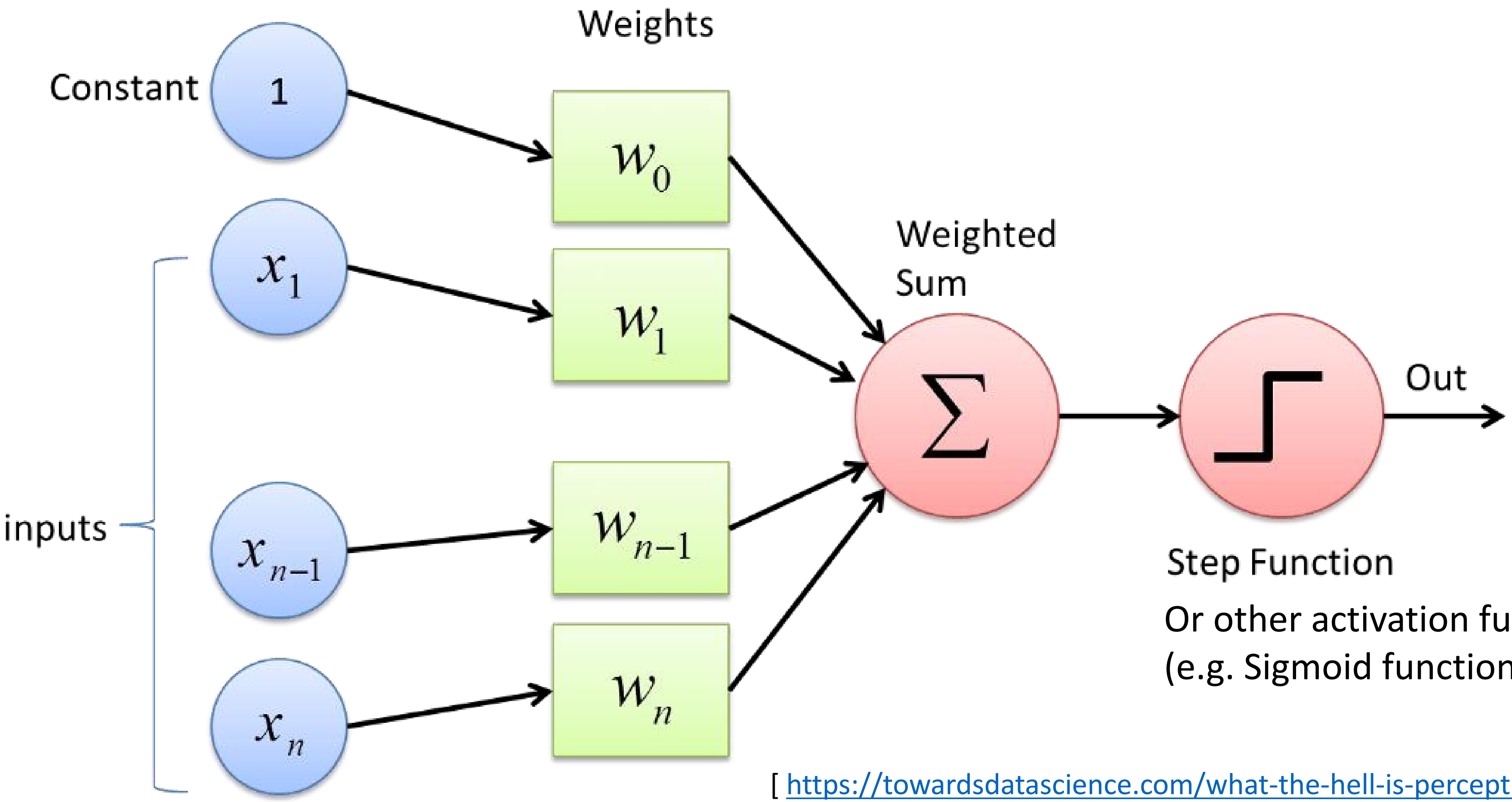
Human brains have millions of neurons; they receive, process, and transmit electric and chemical signals

- Nucleus
- Dendrites: receives signals from other neurons
- Axon: a single output filament
- Synapse: acts as communicator



Source: Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, Pearson.

Artificial Neuron (Perceptron or Single Layer Neural Network)

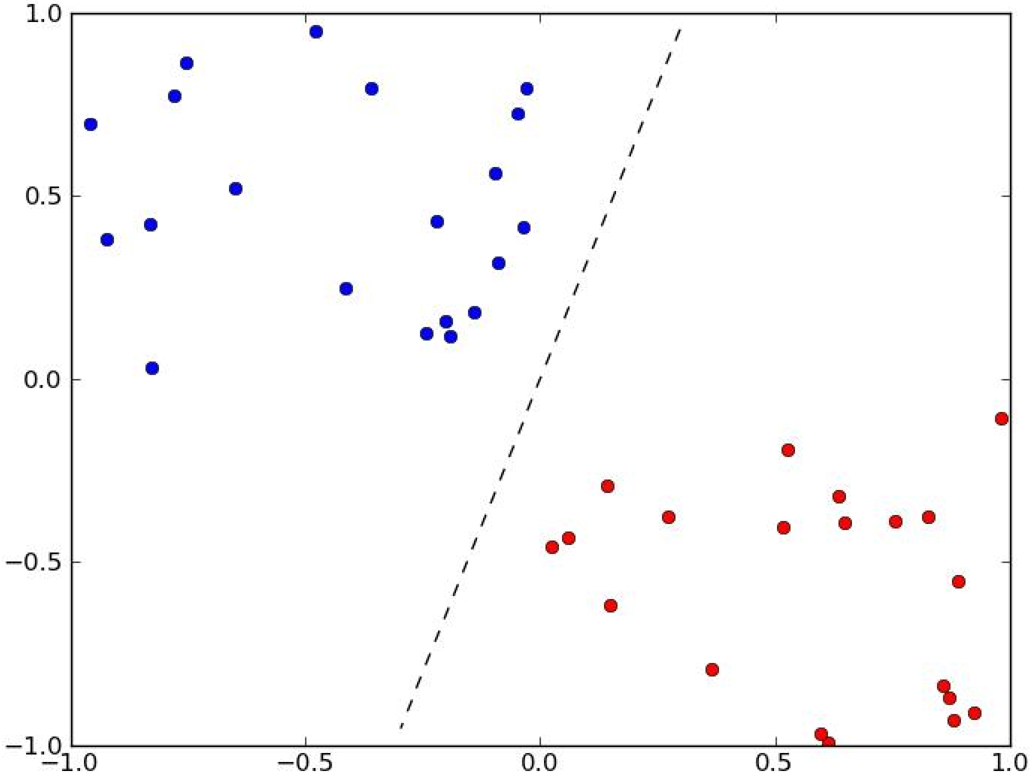


Or other activation function
(e.g. Sigmoid function)

[https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53]

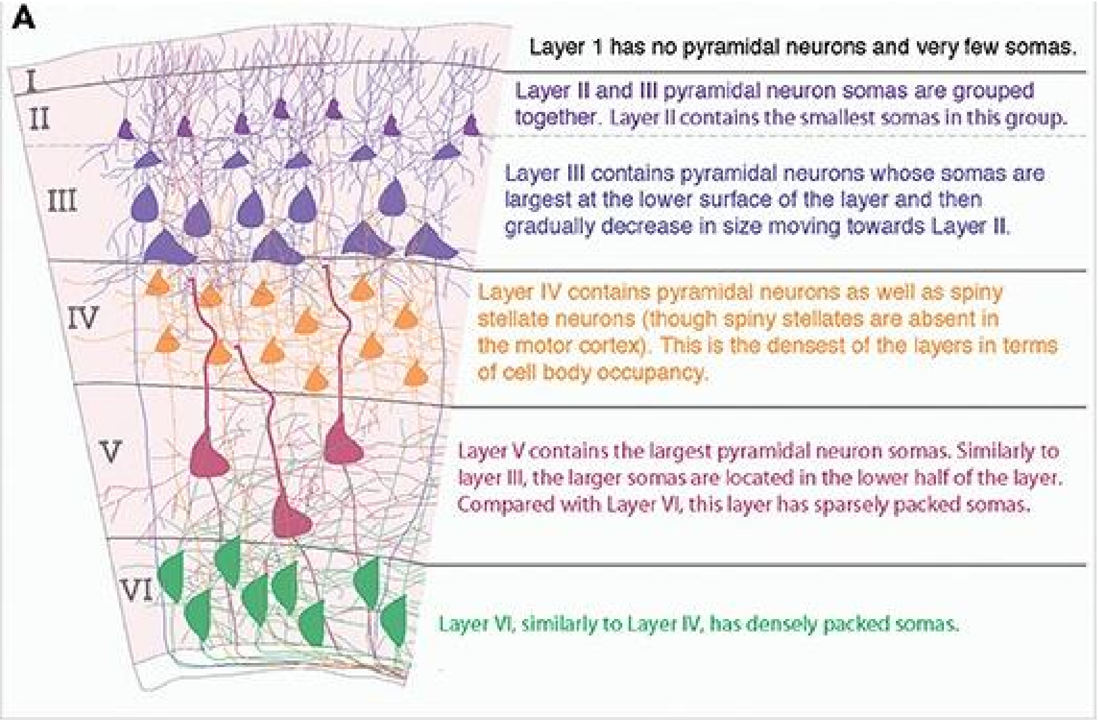
Perceptron for Classification

- Same as logistic regression
- A Perceptron can be trained to perform **binary** classification, when classes are **linearly separable**
- By tuning the weights, we can train the perceptron



Biological Neural Networks

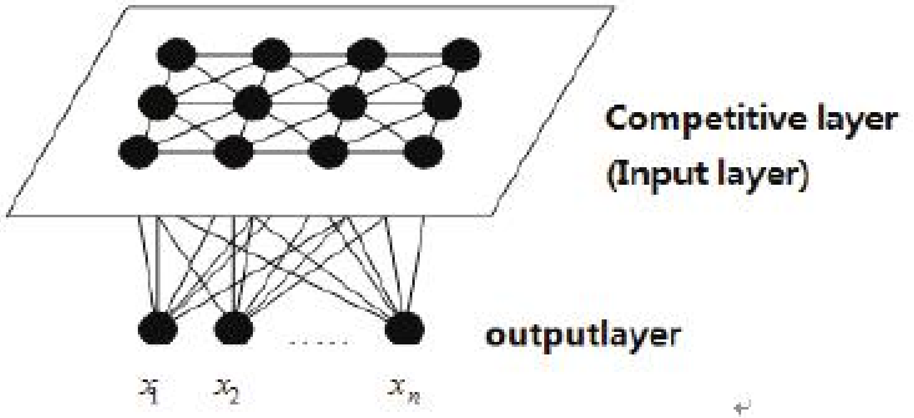
- Cerebral Cortex [Wikipedia]
  - Outer layer of neural tissue of the cerebrum in humans & mammals
  - Key role in attention, perception, awareness, thought, memory, language, and consciousness.
  - Mostly consists of the six-layered neocortex



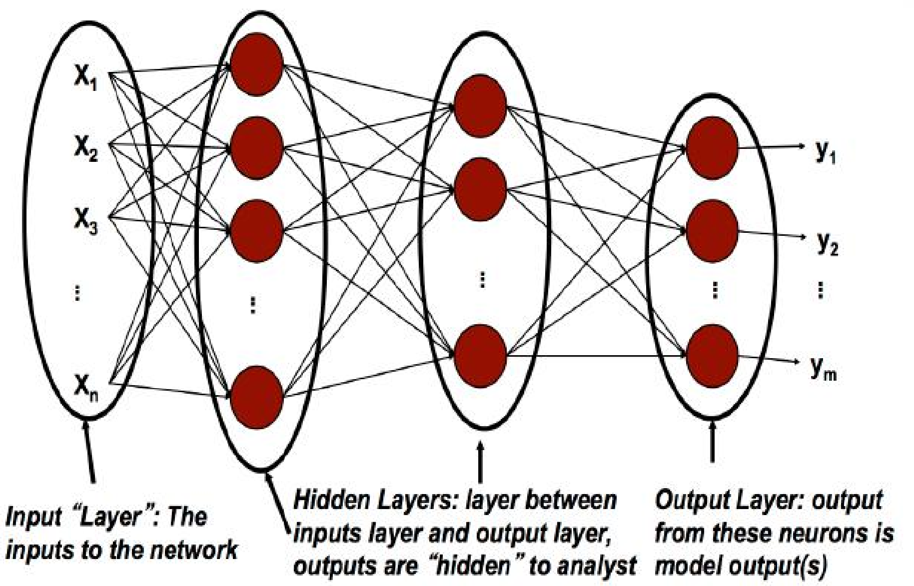
https://www.frontiersin.org/files/Articles/285579/fnana-11-00102-HTML/image_m/fnana-11-00102-g002.jpg

Artificial Neural Networks

[https://www.researchgate.net/figure/The-structure-of-SOM-neural-network_fig1_271637449]



- A network of neurons, connected to each other in a certain way



- Multilayer Perceptron (MLP)
- Most popular structure is **feed-forward** and with **hidden** layers

Learning the Weights

- How do we figure out the weights in the network?
- Training or learning
- Just like a child learning from experience and feedback, a network can be trained
  - Formation and modification of synapses between neurons
  - Trial and error
- Learning algorithms

Neural Net Training- Backpropagation

- First, all weights initialized to small random values
- Loop
  - For all training samples
    - Pass through the network, calculate prediction
    - Calculate the error between prediction and actual target value
    - Back-propagate the error
    - Update weights proportional to the error
- Until convergence to a minimum error

epoch

Issues for Learning in Deep Networks

- **Deep Networks**
  - Having many and many layers

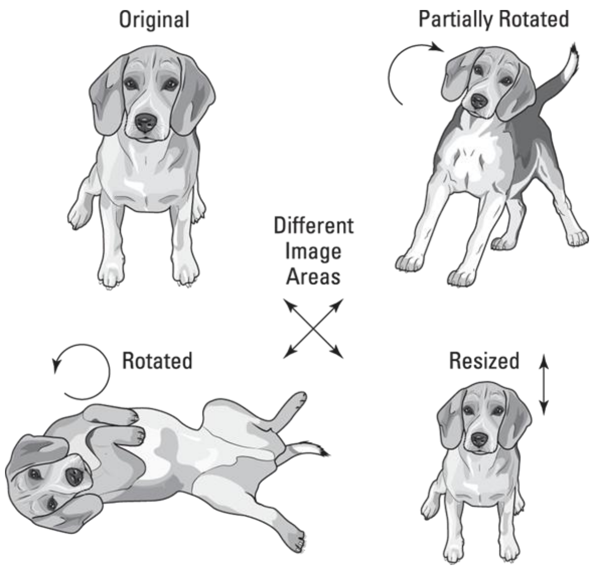
- **Vanishing / Exploding Gradient**
  - When backpropagating through a deep network, the signal quickly fades to near zero values
  - Activation functions need large enough values to let the signal pass; therefore, low values are blocked.
  - The farther neuron layers are from the output, the higher the likelihood that they'll get locked out of updates.
  - Therefore, the network stops learning as a whole (or learns at incredibly slow pace).

Issues for Learning in Deep Networks (cont.)

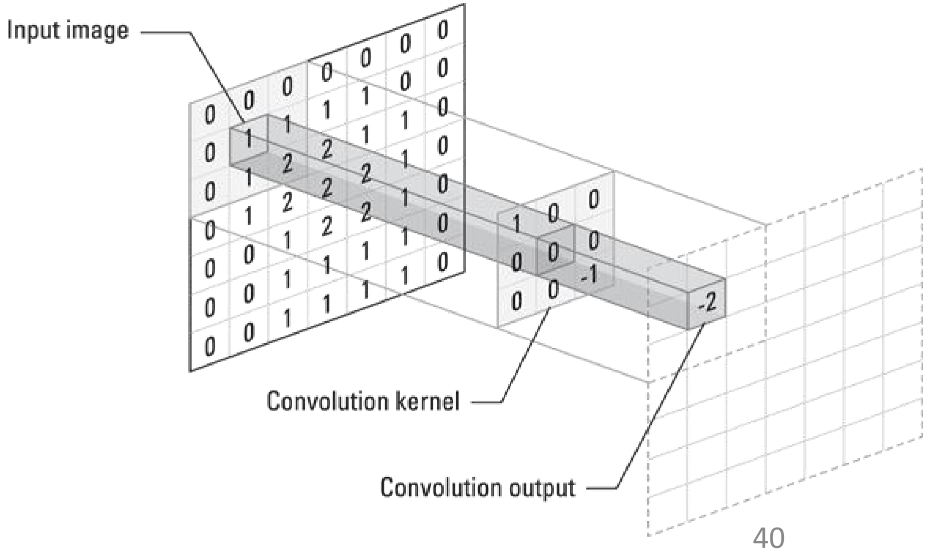
- New improvements in Neural Net training, overcoming previous shortcomings, that allows for training of **deep** networks.
  - Improvements in algorithm
  - Geoffrey Hinton, UofT (now Google)
  - ImageNet Classification with Deep Convolutional Neural Networks: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton 2012

Deep Convolutional Neural Nets (CNN)

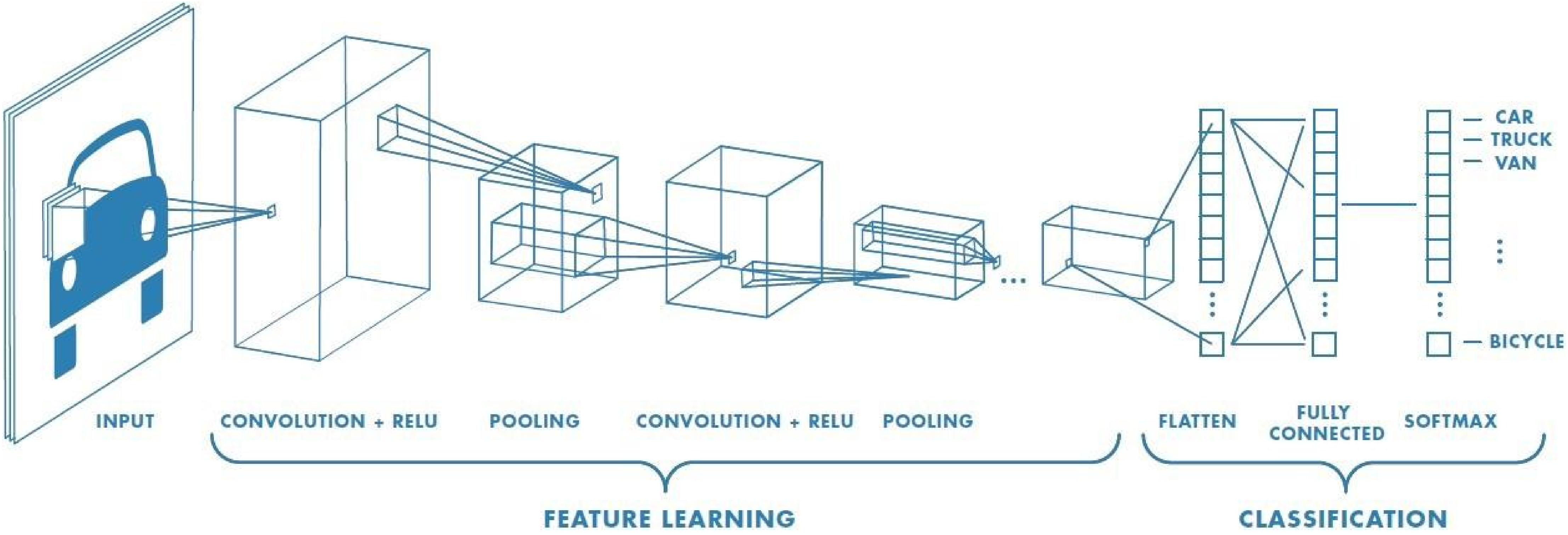
- Invariance
  - Having the same response regardless of size, distortion, or position in the image



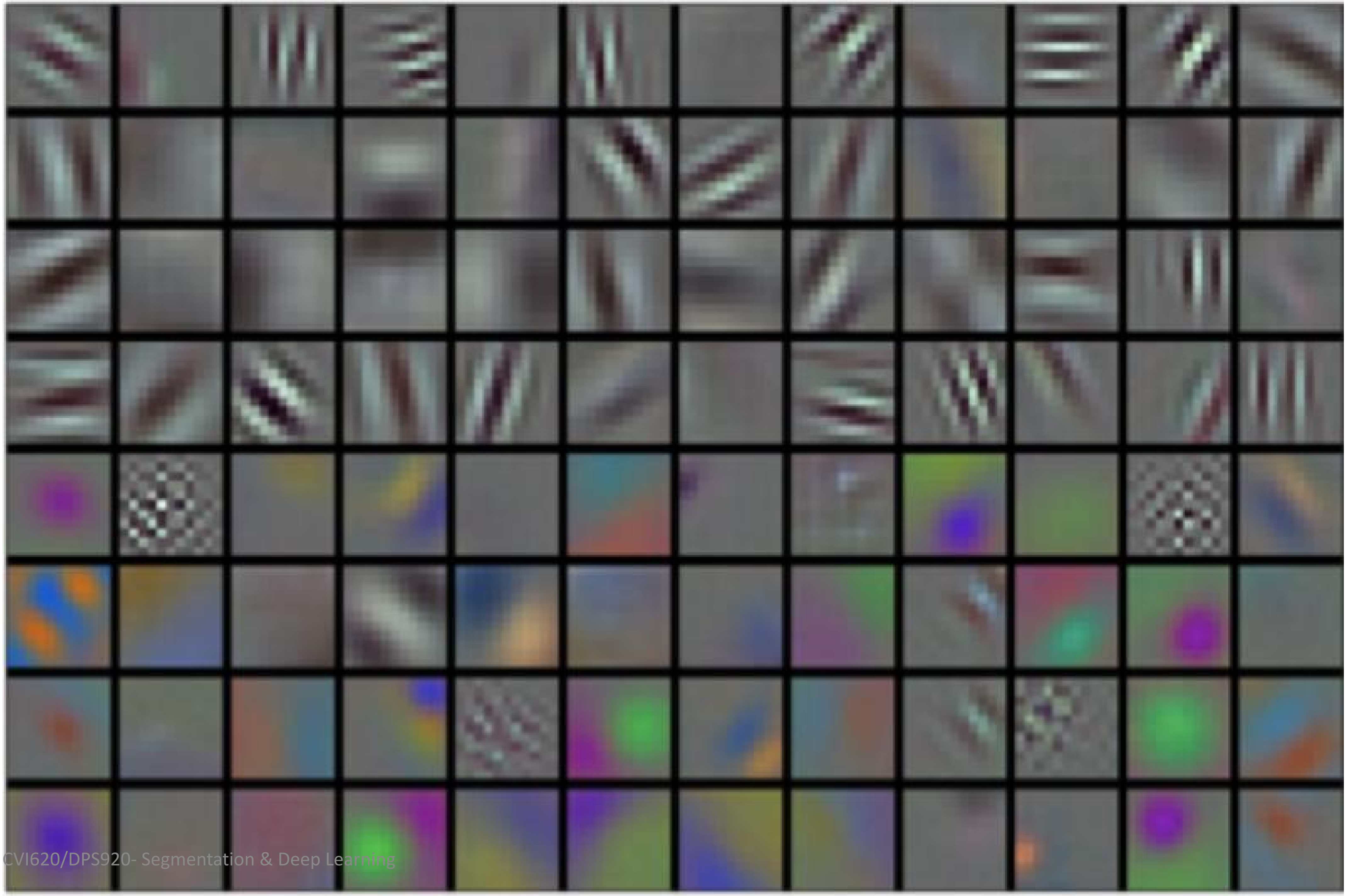
- Based on “Convolution”
  - Applied similarly to all parts (tiles) of images



Example

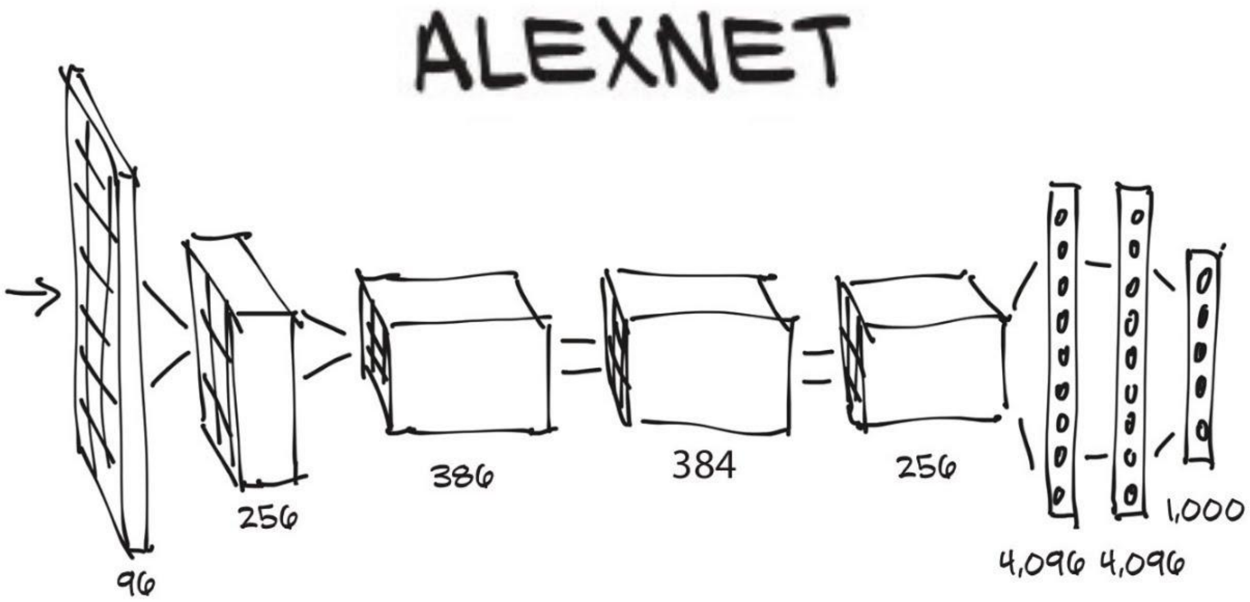


https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53



Open-Source Frameworks

- Share the **trained** neural nets
  - NN Architecture
  - Learned weights
- Democratization of Deep Learning models

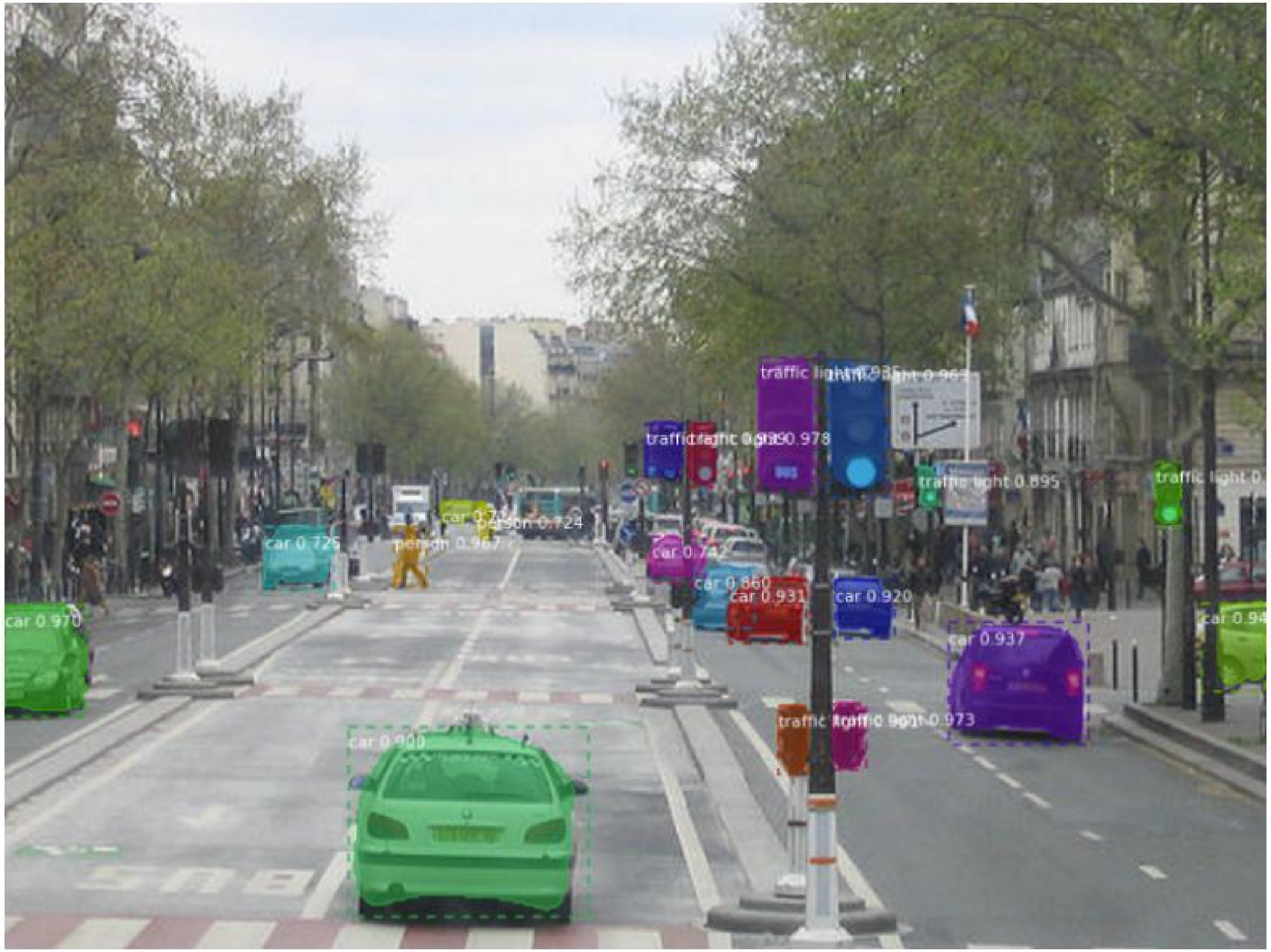


YOLO V8 [https://yolov8.com/]

Classification	Detection	Segmentation
		

https://www.freecodecamp.org/news/how-to-detect-objects-in-images-using-yolov8/

MASK R-CNN [https://github.com/matterport/Mask_RCNN]



GoogLeNet (Inception) [https://www.kaggle.com/keras/inceptionv3/home]

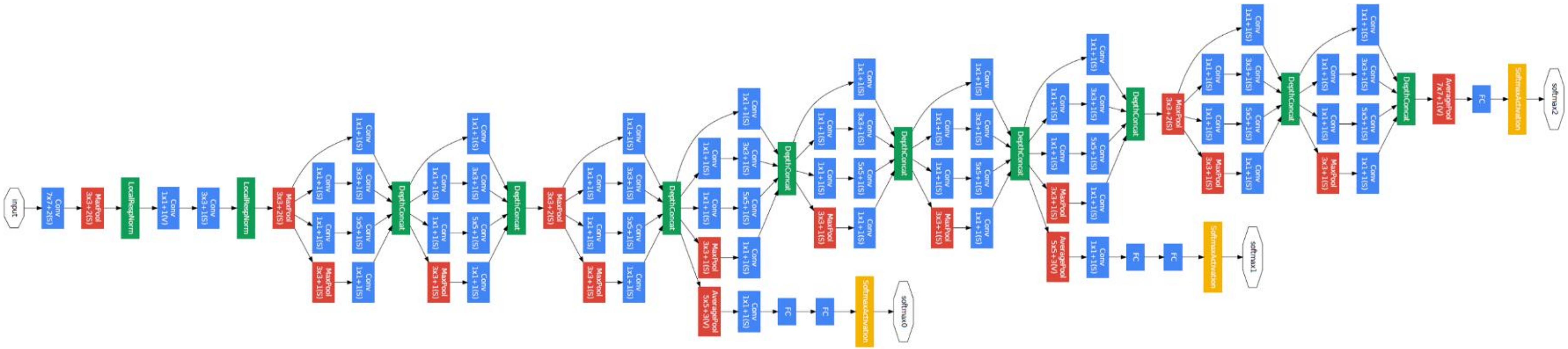
Image

Inception Labels

1. **Shih-Tzu: 79.418%**
2. Lhasa: 2.359%
3. Affenpinscher: 1.567%
4. Pekinese: 0.954%
5. Tibetan Terrier: 0.661%

1. **Assault Rifle: 87.250%**
2. Rifle: 6.149%
3. Bulletproof Vest: 3.534%
4. Military Uniform: 1.273%
5. Stretcher: 0.060%

Inception V1

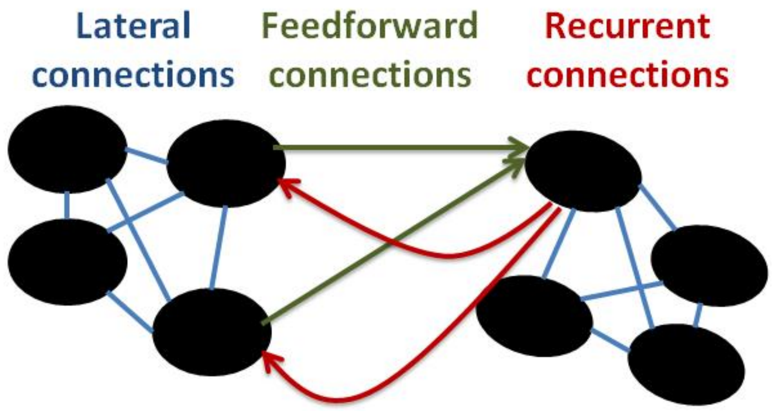


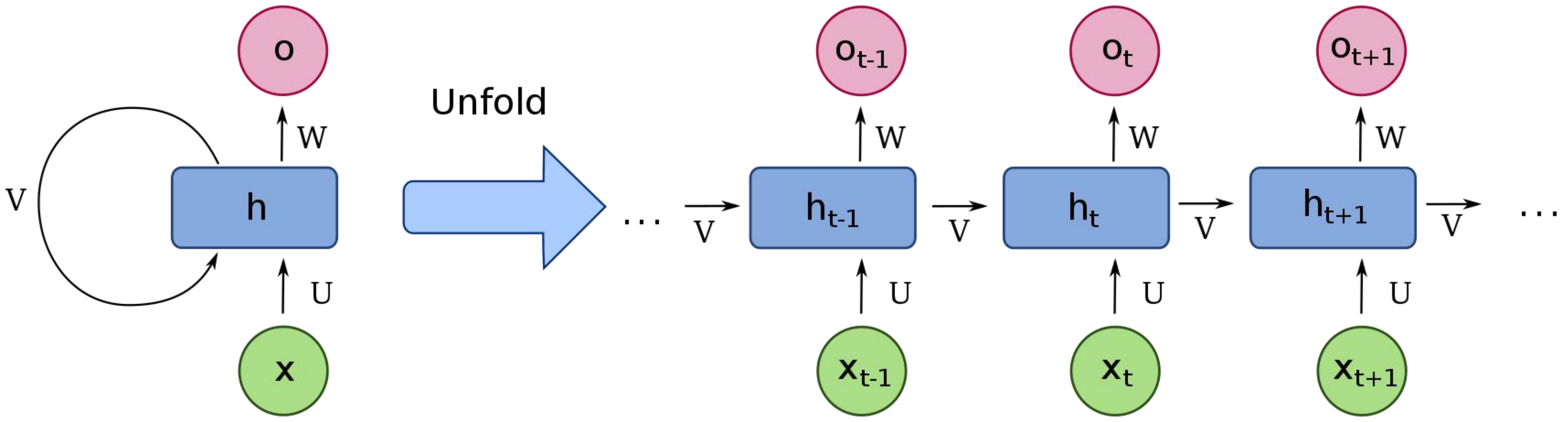
https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch/

- https://learnopencv.com/ultralytics-yolov8/
- https://jonascleveland.com/best-image-classification-models/
- VGG-Net (Visual Geometry Group) https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f

Deep Recurrent Neural Nets (RNN)

- Temporal sequence
- For example, for processing video or understanding natural language (e.g. English sentences)





[Recurrent neural network – Wikipedia]

Transfer Learning

- But… Training DL models to solve new problems is not easy. Training Neural Nets requires A LOT of labelled data.
  - Need big data and computing power
  - Access to lots of computing power (usually GPU processing)

- Good news!
  - “Extending a network’s capability to new kinds of images that weren’t part of the previous learning means transferring the knowledge to this new problem (transfer learning).”

  - “Use the majority of the layers of the network as they are (you freeze them) and then work on the final, output layers (**fine-tuning**)”
    - For example, use InceptionV3 feature extraction, but learn a new set of categories

Summary

- **Image segmentation** is the task of finding groups of pixels that "go together".

- In salient **object segmentation**, the goal is to group pixels that belong to the same object

- Many methods exist for segmentation, each with its strengths and weaknesses. Recent methods based on Deep Learning have a good performance.

Summary (cont.)

- Computer Vision problems can be solved using three approaches: 1) Start from intuition and design a method, or 2) Start from data and use Machine Learning (ML) 3) Start from LOTS of data and use **Deep learning,** a new ML technique inspired by biological brains.

- Artificial Neural Networks solve AI / Machine Learning problems by building an artificial “brain” made of artificial neurons.

- An artificial neuron or **perceptron** can be trained to perform linearly separable binary classification. However, interesting problems require complex artificial neural networks often in a deep (multi-layer) architectures.

Summary (cont.)

- Until recently, training these networks was the challenge.
- Deep Learning methods offer improvements to **backpropagation** training and remove issues such as the **vanishing gradient** problem.
- Trained neural networks, such as **CNNs** for object detection in images, or **RNNs** for natural language processing, are often shared. These networks can be further trained or **fine-tuned** for customization.
- With the success of Deep Learning methods in training Neural Nets, many problems in ML and Computer Vision were solved!
- One specific architecture, called Convolutional Neural Net (CNN), implements convolution (or filtering) and thus feature detection, and is very effective in solving object detection, among other things.

References

[1] Computer Vision: Algorithms and Applications, R. Szeliski (http://szeliski.org/Book)

[2] Learning OpenCV 3, A. Kaehler & G. Bradski

- Available online through Safari Books, Seneca libraries
- https://senecacollege-primo.hosted.exlibrisgroup.com/primo-explore/fulldisplay?docid=01SENC_ALMA5153244920003226&context=L&vid=01SENC&search_scope=default_scope&tab=default_tab&lang=en_US

[3] Practical introduction to Computer Vision with OpenCV, Kenneth Dawson-Howe

- Available through Seneca libraries
- https://senecacollege-primo.hosted.exlibrisgroup.com/primo-explore/fulldisplay?docid=01SENC_ALMA5142810950003226&context=L&vid=01SENC&search_scope=default_scope&tab=default_tab&lang=en_US

References (cont.)

[4] Movahedi, V., “Automatic Extraction of Closed Contours Bounding Salient Objects: New Algorithms and Evaluation Methods”, Ph.D. Dissertation, York University, 2015.

[5] Howse, J., “Learning OpenCV 4 Computer Vision with Python 3”, Packt Publishing, 2020.

Readings

- Chapter 4 [2]
- Chapter 12 [3]

- [1] **A Practical Introduction to Computer Vision with OpenCV** by Kenneth Dawson-Howe Available online via library.senecacollege.ca: Permalink
- [2] **Learning OpenCV 4 Computer Vision with Python 3** by J. Howse & J. Minichino Available online via library.senecacollege.ca: Permalink
- [3] **Learning OpenCV 3** by A. Kaehler & G. Bradski Available online via library.senecacollege.ca: Permalink [4] **Mastering OpenCV with practical computer vision projects** by D.L. Baggio et al. Available online via library.senecacollege.ca: Permalink

Blobs

- A group of connected pixels in an image, sharing some property, e.g. grayscale or color

OpenCV SimpleBlobDetector

```
using namespace cv;
// Read image
Mat im = imread("blob.jpg", IMREAD_GRAYSCALE);

// Set up the detector with default parameters.
SimpleBlobDetector detector;

// Detect blobs.
std::vector<KeyPoint> keypoints;
detector.detect(im, keypoints);

// Draw detected blobs as red circles.
// DrawMatchesFlags::DRAW_RICH_KEYPOINTS flag ensures the size of the circle corresponds to the
size of blob
Mat im_with_keypoints;
drawKeypoints(im, keypoints, im_with_keypoints, Scalar(0, 0, 255),
DrawMatchesFlags::DRAW_RICH_KEYPOINTS);

// Show blobs
imshow("keypoints", im_with_keypoints);
waitKey(0);
```

Older versio n?

Use :
https://raw.githubusercontent.com/opencv/opencv/master/samples/cpp/detect_blob.cpp

https://stackoverflow.com/questions/30622304/opencv-3-blobdetection-the-function-feature-is-not-implemented-in-detectand

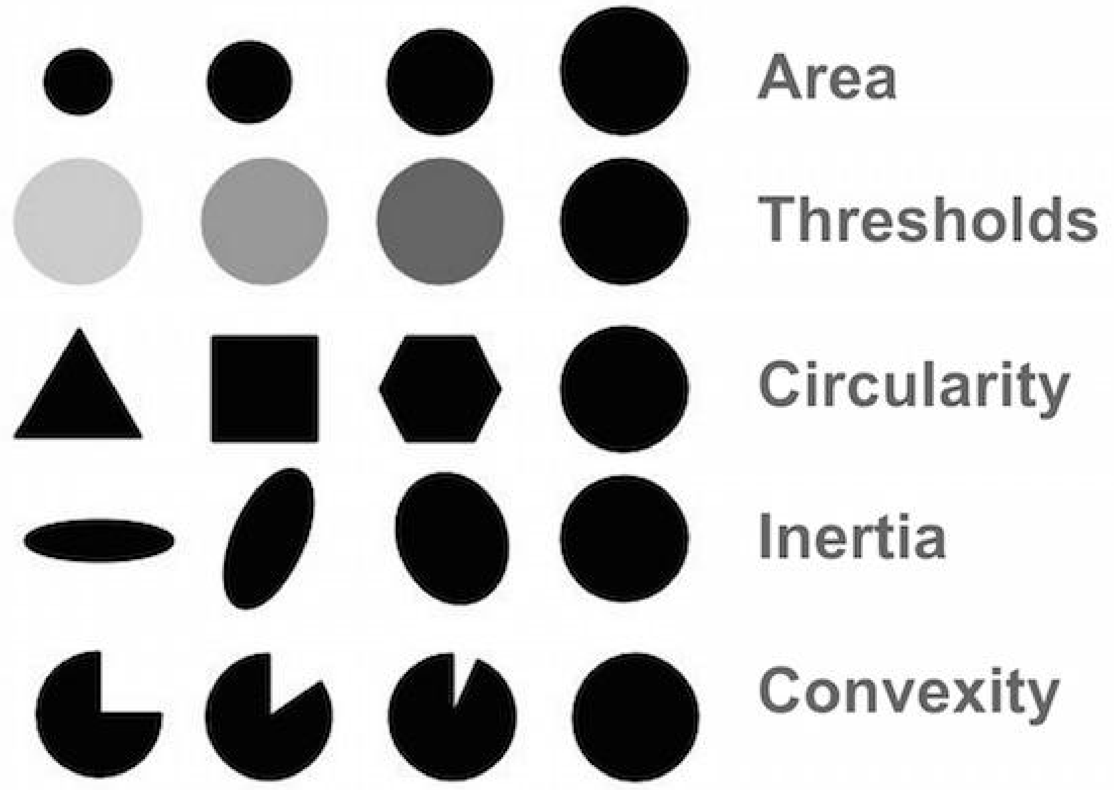
[https://www.learnopencv.com/blob-detection-using-opencv-python-c/]

This class performs several filtrations of returned blobs. You should set `filterBy*` to true/false to turn on/off corresponding filtration. Available filtrations:

- **By color**. This filter compares the intensity of a binary image at the center of a blob to `blobColor`. If they differ, the blob is filtered out. Use `blobColor = 0` to extract dark blobs and `blobColor = 255` to extract light blobs.
- **By area**. Extracted blobs have an area between `minArea` (inclusive) and `maxArea` (exclusive).
- **By circularity**. Extracted blobs have circularity ($\frac{4*\pi*Area}{perimeter*perimeter}$) between `minCircularity` (inclusive) and `maxCircularity` (exclusive).
- **By ratio of the minimum inertia to maximum inertia**. Extracted blobs have this ratio between `minInertiaRatio` (inclusive) and `maxInertiaRatio` (exclusive).
- **By convexity**. Extracted blobs have convexity (area / area of blob convex hull) between `minConvexity` (inclusive) and `maxConvexity` (exclusive).

Default values of parameters are tuned to extract dark circular blobs.

[https://docs.opencv.org/3.0-last-rst/modules/features2d/doc/common_interfaces_of_feature_detectors.html#simpleblobdetector]



[https://www.learnopencv.com/blob-detection-using-opencv-python-c/]

Regional methods

Homogeneity Measure

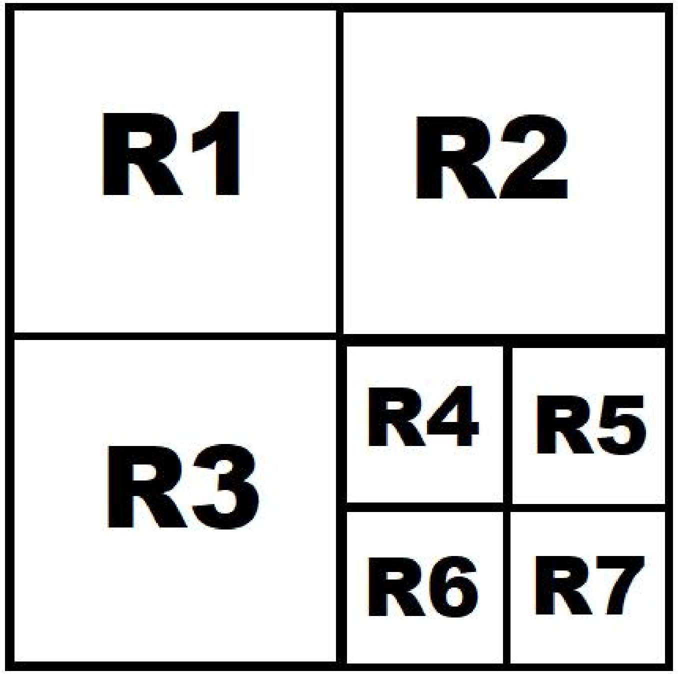
- How do you measure if a region contains similar pixels?
  - Can use brightness, color, texture, etc.
  - Example: gray-level variance

1. Define a homogeneity measure (H)
2. Set a threshold (t)

- If for a region R, H(R) > t , then pixels are similar, and belong to same segment
- Otherwise, pixels do not belong to the same segment

Split-only Segmentation

- Start from the whole image as one region
- Loop over all (new) regions
  1. Calculate the **homogeneity** measure
  2. If the measure < threshold (did not pass the test), split the region (e.g. into 4 regions)
- Continue until all regions pass the homogeneity test



Example: Merge-only

- Start from the lowest level (e.g. each pixel being a region or each NxN neighborhood as a region)
- Loop over all (new) regions
  1. Try merging the region with neighbors
  2. Calculate the homogeneity measure
  3. Keep the merged region, if the measure > threshold (passing the test)

Split & Merge Techniques[5]

- Procedure:
  1. Split the image into equally sized regions
  2. Calculate the homogeneity measure for each region
  3. If the measure > threshold (passed the test), attempt to merge with its neighbor(s). If not, split the region
  4. Continue until all regions pass the homogeneity test

FloodFill flags

- Possible values for flag:

Low 8 bits	4	use 4-connectivity
	8	use 8-connectivity
High 8 bits	cv::FLOODFILL_FIXED_RANGE	A pixel will be compared to the original seed rather than to its neighbors
	cv::FLOOD_MASK_ONLY	The input image will not be modified, instead a mask argument will be the output (see [2] page 362)
Middle 8 bits	A numerical value	The value to use when filling the mask (default is filling with 1s)

- Use OR to set multiple flags:

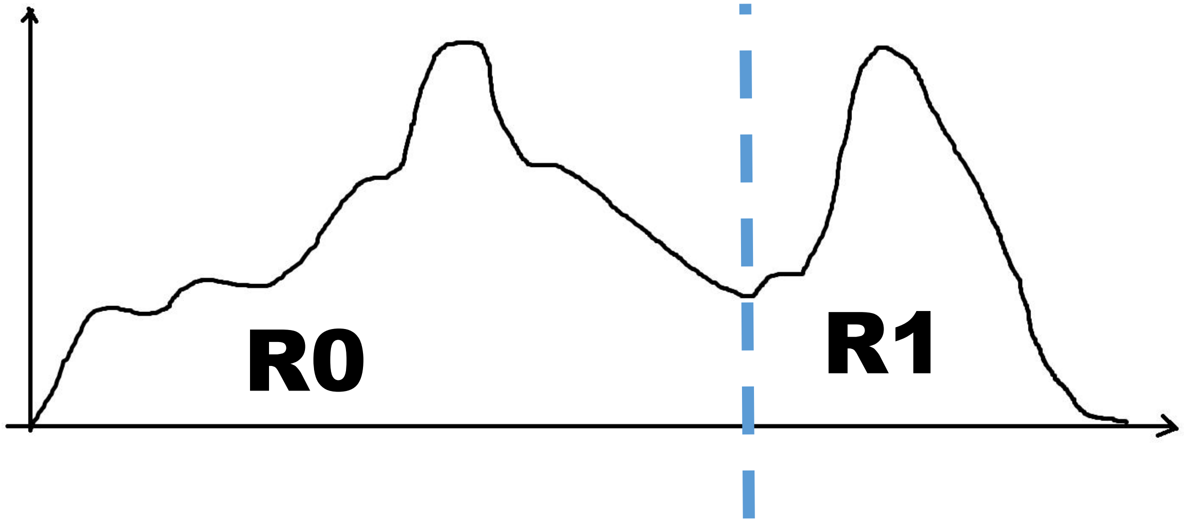
```
flags = 8
      | cv::FLOODFILL_MASK_ONLY
      | cv::FLOODFILL_FIXED_RANGE
      | (47<<8);
```

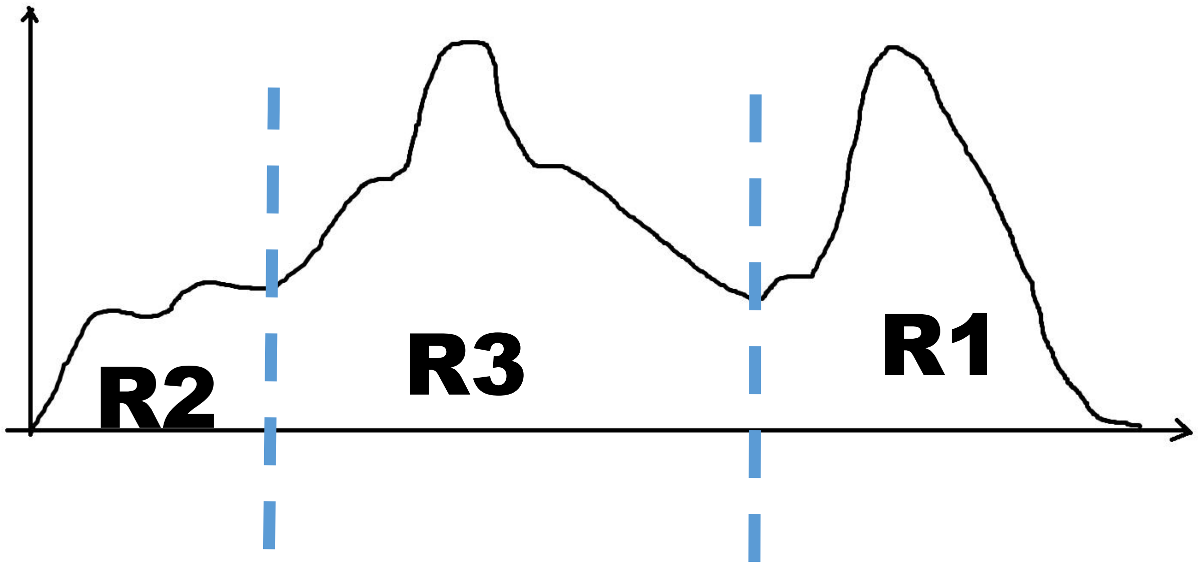
Clustering methods

- Similar ideas of splitting and merging, but not applied in the spatial (row and column) domain
- Instead other domains, such as the histogram
- Therefore can group pixels that are not connected
- Suitable for **occlusion** (when an object is partially covered)

Histogram-based methods

- Simple example:
  - Start as the entire image as one region
  - Loop until reaching the expected number of regions
    - Find a peak and put thresholds on either side of the peak
    - Use the thresholds to split the region into two regions





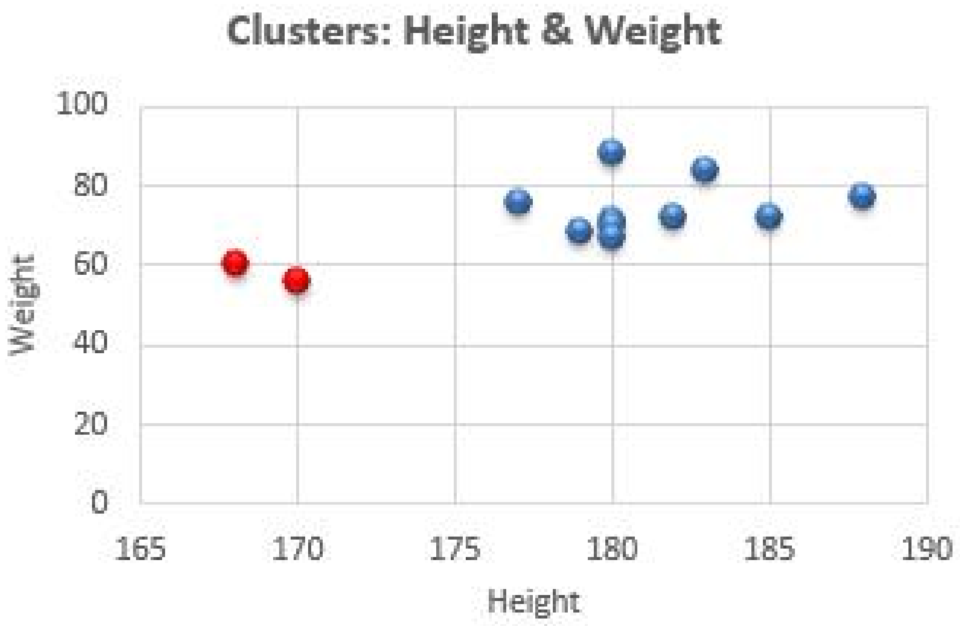
K-means clustering

- A method for clustering (grouping) data into *k* groups
- Needs a measure of similarity/ distance between two data points
  - Examples:
    - Color similarity between two pixels
    - Spatial distance between two pixels
    - Or more complex measures
- Procedure
  - Initialize k centers or means (m_1, m_2, …, m_k)
  - Loop until no change
    - *Assignment step*: Assign all data to the most similar/ nearest center
    - *Update step*: Calculate the new center/mean as the mean of data belonging to the same cluster

https://youtu.be/4b5d3muPQmA

Example- 2 dimensional data

- [http://dni-institute.in/blogs/k-means-clustering-algorithm-explained/]



Other segmentation methods

Graph methods

- Regions are modelled as the nodes in a graph.
- Neighboring regions are connected by edges in the graph
- The weights of the edges is defined by the similarity between the regions
- Using graph optimization methods to group the nodes
- For example:
  - Graph-cut
  - Min-cut
  - Grab-cut

Segmentation Algorithms [4]

- Regional methods
  - Optimize the labelling of pixels
- Active Contour methods
  - Find the optimal deformation and location of a deformable contour around an object in the image
- Contour grouping methods
  - Search for the optimal grouping of boundary entities (e.g. edgels, line segments or curvelets) to form an object contour
- Deep Learning methods
  - Using Neural Networks and training them using lots of data

OpenCV findContours

findContours

Finds contours in a binary image.

C++: void **findContours**(InputOutputArray **image**, OutputArrayOfArrays **contours**, OutputArray **hierarchy**, int **mode**, int **method**, Point **offset**=Point())

C++: void **findContours**(InputOutputArray **image**, OutputArrayOfArrays **contours**, int **mode**, int **method**, Point **offset**=Point())

[https://docs.opencv.org/3.0-last-rst/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html]

```cpp
namedWindow( "image", 1 );
imshow( "image", img );
//Extract the contours so that
vector<vector<Point> > contours0;
findContours( img, contours0, hierarchy, RETR_TREE, CHAIN_APPROX_SIMPLE);

contours.resize(contours0.size());
for( size_t k = 0; k < contours0.size(); k++ )
    approxPolyDP(Mat(contours0[k]), contours[k], 3, true);
```

```cpp
drawContours( cnt_img, contours, _levels <= 0 ? 3 : -1, Scalar(128,255,255),
              3, LINE_AA, hierarchy, std::abs(_levels) );

imshow("contours", cnt_img);
```

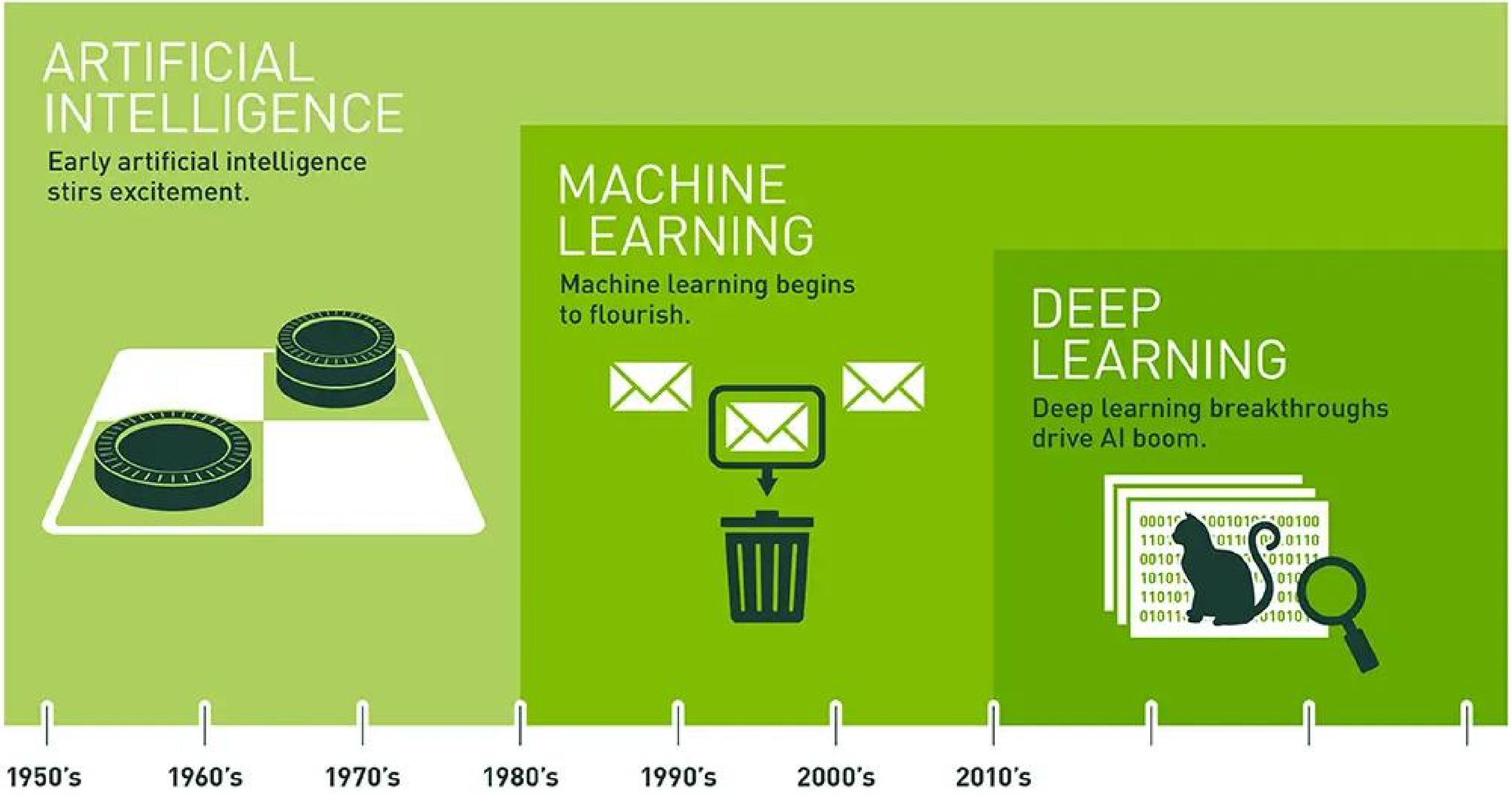
[https://github.com/opencv/opencv/blob/master/samples/cpp/contours2.cpp]

Iterative training & epochs

- Backpropagation
  - Mathematical formulation for using feedback from trial and error to train the network
- **Epochs**: Repeated iterations of seeing training samples

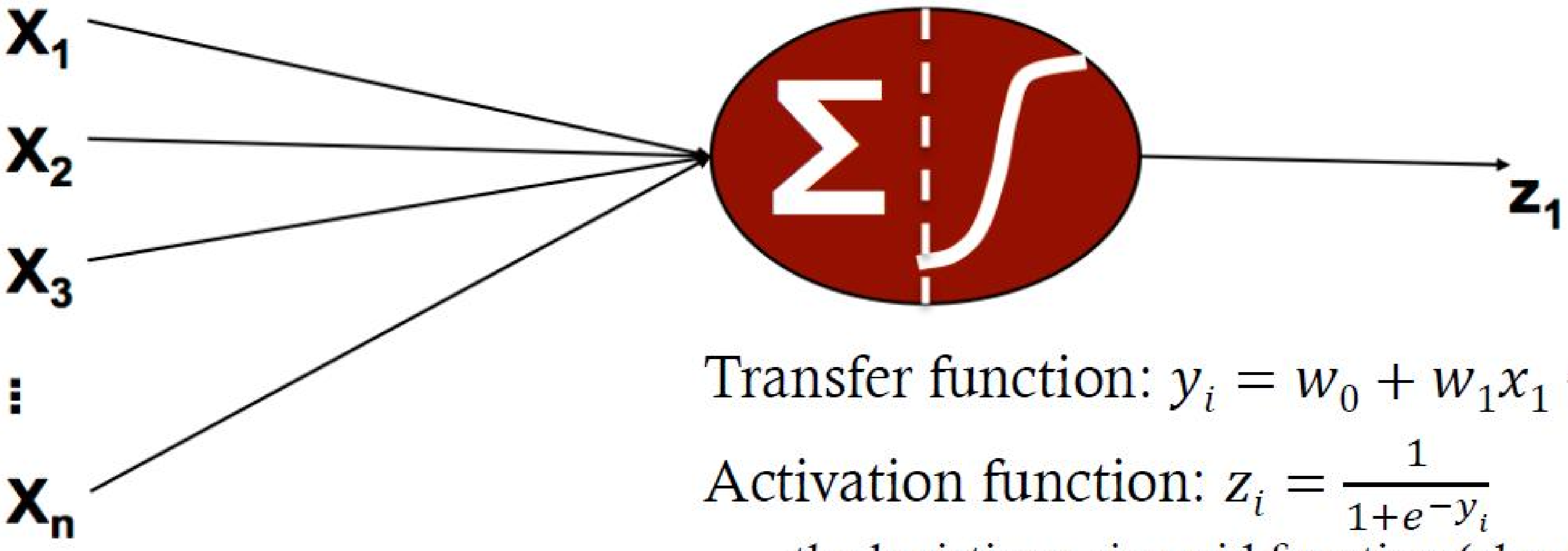
- Learn from mistakes!
- The **error** calculated at the last layer is sent back through the network and used for calculating adjusted weights

Timeline



[https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/]

Artificial Neuron



Transfer function: $y_i = w_0 + w_1 x_1 + \cdots + w_n x_n$

Activation function: $z_i = \frac{1}{1+e^{-y_i}}$

the logistic or sigmoid function (above)

Or other squashing functions

Single Neuron is a linearly weighted sum, followed by a "squashing" function. z_1 has range [0,1]

Computing for Deep Learning

- Huge amounts of data needed for training deep nets
- Many iterations (epochs) needed
- Not feasible using normal computers

- Improvements in Computing
  - GPU computing for images
  - Cloud/clusters (Google, Facebook, Microsoft, IBM)

- Example
  - Google Brain project, 16,000 computers, more than a billion weights, for unsupervised learning from YouTube videos

Examples of DL models & applications

- Image classification, object detection
- Visual Object Tracking
- Automatic Speech Recognition
- Natural Language Processing
- Recommendation
  - Netflix
  - Amazon

Limitations of DL

- Needs lots and lots of data
- If supervised learning, all need to be labelled
- Needs lots of computing power/ time

- DL models do not generalize
  - If only seen samples of cats and dogs, they won't (necessarily) generalize to learn a class of animals

- Not easy to explain their output (how and why)

Examples

- Deep Learning In 5 Minutes: https://youtu.be/6M5VXKLf4D4
- Top Deep Learning Projects: https://youtu.be/N082bM72byY
- 9 Cool Deep Learning Applications: https://youtu.be/Bui3DWs02h4
- 10 More Cool Deep Learning Applications: https://youtu.be/hPKJBXkyTKM