A Overview of Computer Vision Tasks, including Multiple-Object Detection (MOT)

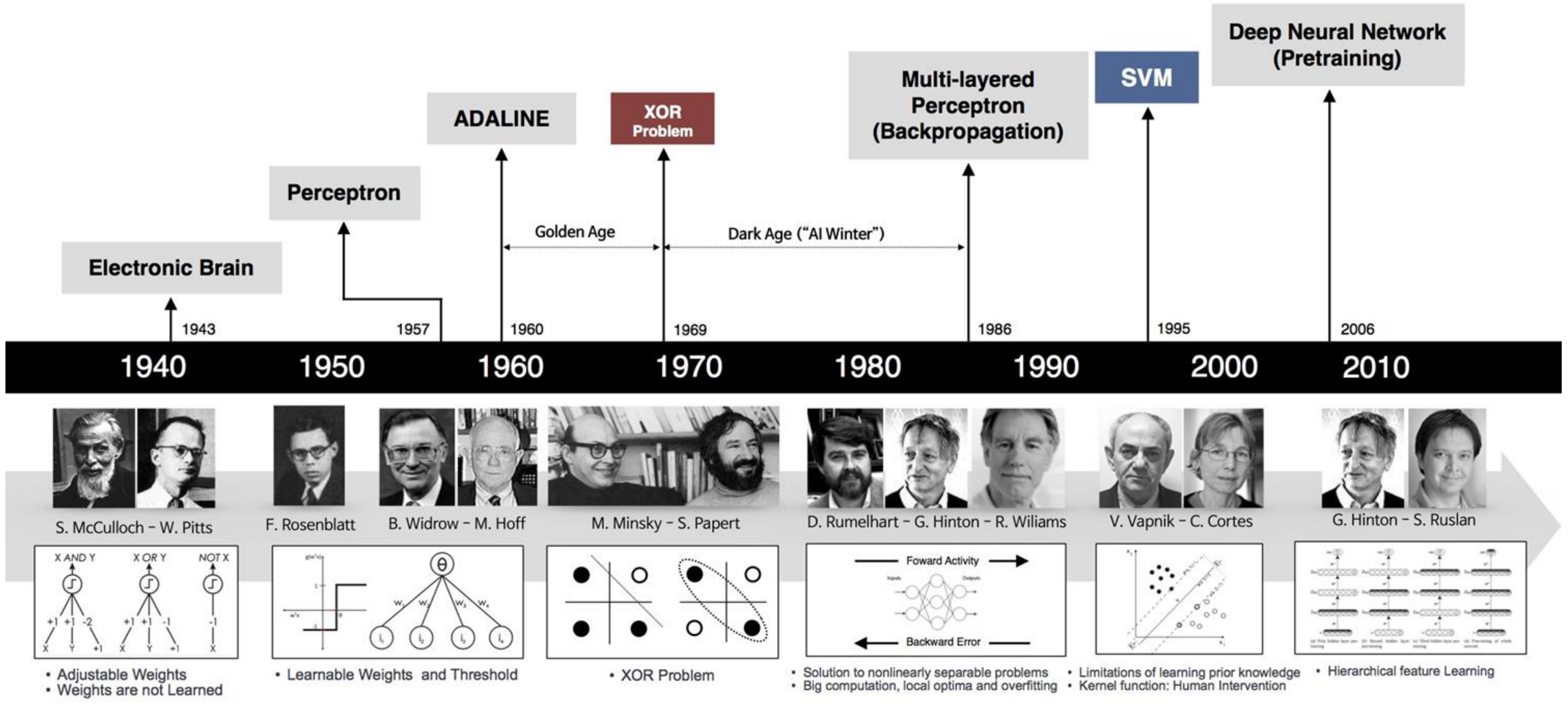
Anthony D. Rhodes

5/2018

Contents

A Very Brief Introduction to Neural Networks and Deep Learning

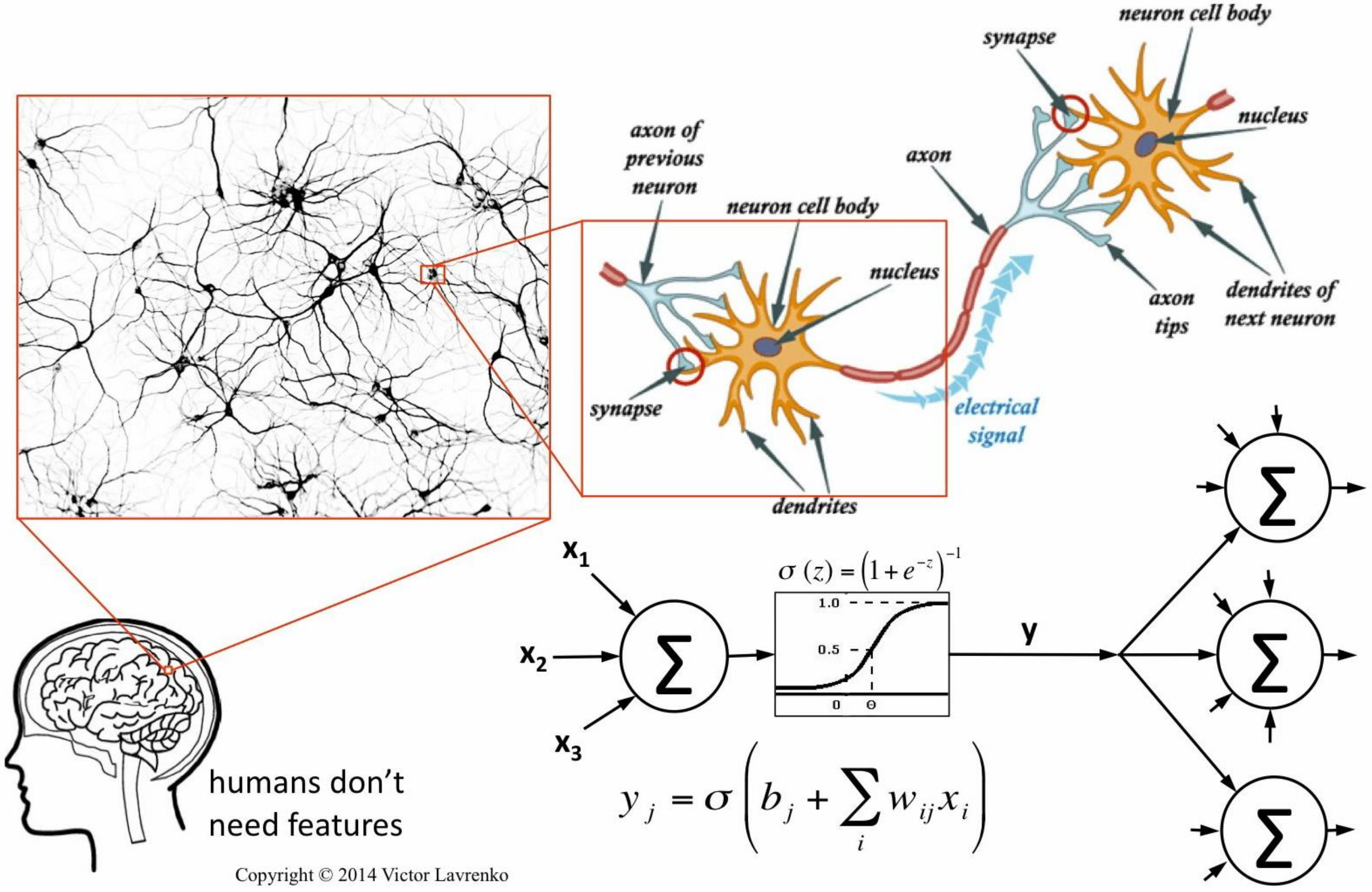
A Bit of History


Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
XOR Problem
ADALINE
Perceptron
Electronic Brain
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

(*) End of "AI Winter" comes about with the rediscovery of the "backpropagation" algorithm; LeCun et al., (1998) error rate < 1% on MNIST (handwritten digit recognition task).

(*) Inception of "Deep Learning" era begins with landmark "Alexnet" architecture: Krizehevsky et al., (one of the first use of GPUs, among other innovations).

Neurons & the Brain

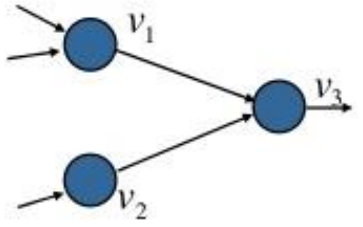


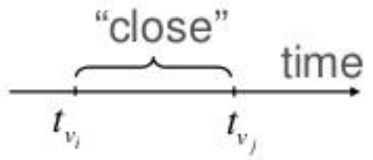
Hebb’s Postulate

“When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.”

- In other words: if two neurons fire “close in time” then strength of synaptic connection between them increases.

$$\Delta w_{ij}(t) = \eta \nu_i \nu_j g(t_{\nu_i}, t_{\nu_j})$$

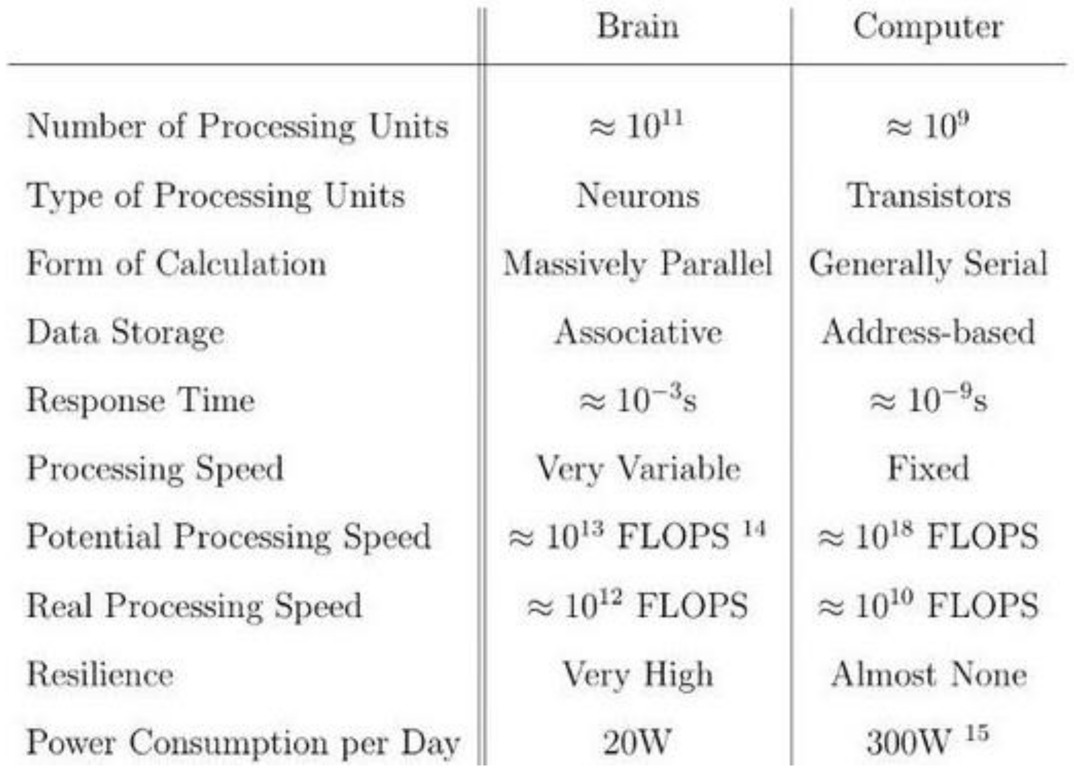




- Weights reflect correlation between firing events.

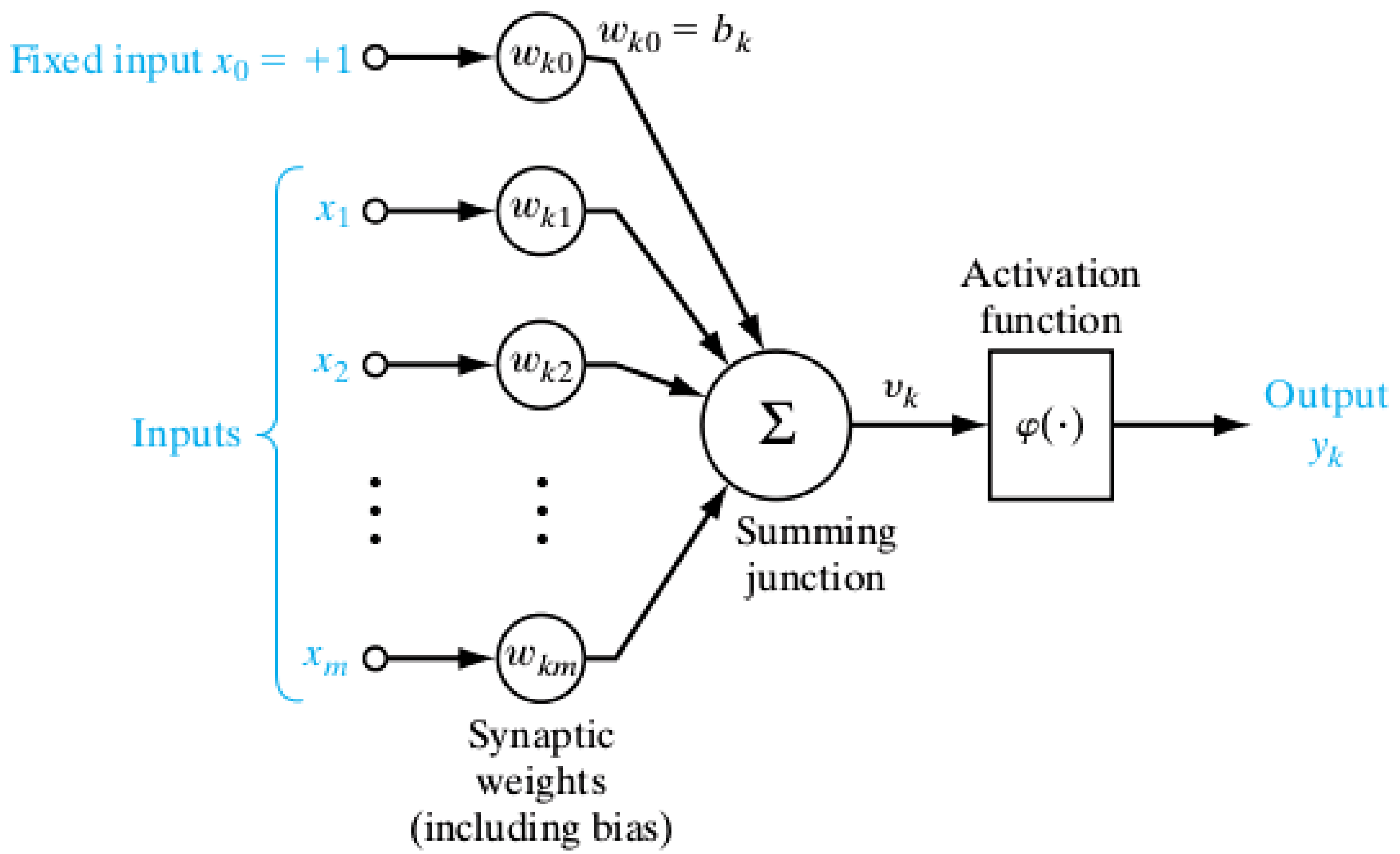
Neurons & the Brain

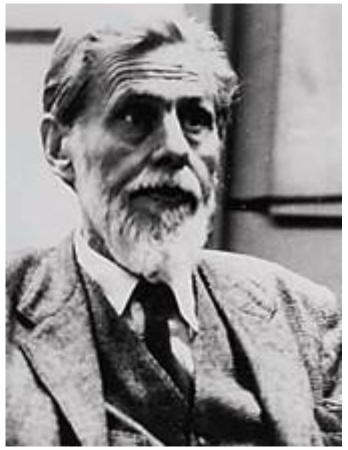
- Human brain contains ~10^{11} neurons
- Each individual neuron connects to ~10^{4} neuron
- ~10^{14} total synapses!

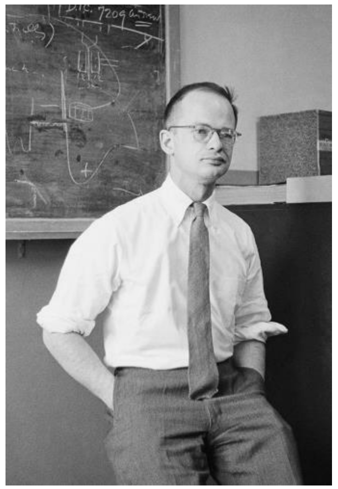


	Brain	Computer
Number of Processing Units	$\approx 10^{11}$	$\approx 10^{9}$
Type of Processing Units	Neurons	Transistors
Form of Calculation	Massively Parallel	Generally Serial
Data Storage	Associative	Address-based
Response Time	$\approx 10^{-3}$s	$\approx 10^{-9}$s
Processing Speed	Very Variable	Fixed
Potential Processing Speed	$\approx 10^{13}$ FLOPS [14]	$\approx 10^{18}$ FLOPS
Real Processing Speed	$\approx 10^{12}$ FLOPS	$\approx 10^{10}$ FLOPS
Resilience	Very High	Almost None
Power Consumption per Day	20W	300W [15]

McCulloch & Pitts Neuron Model (1943)



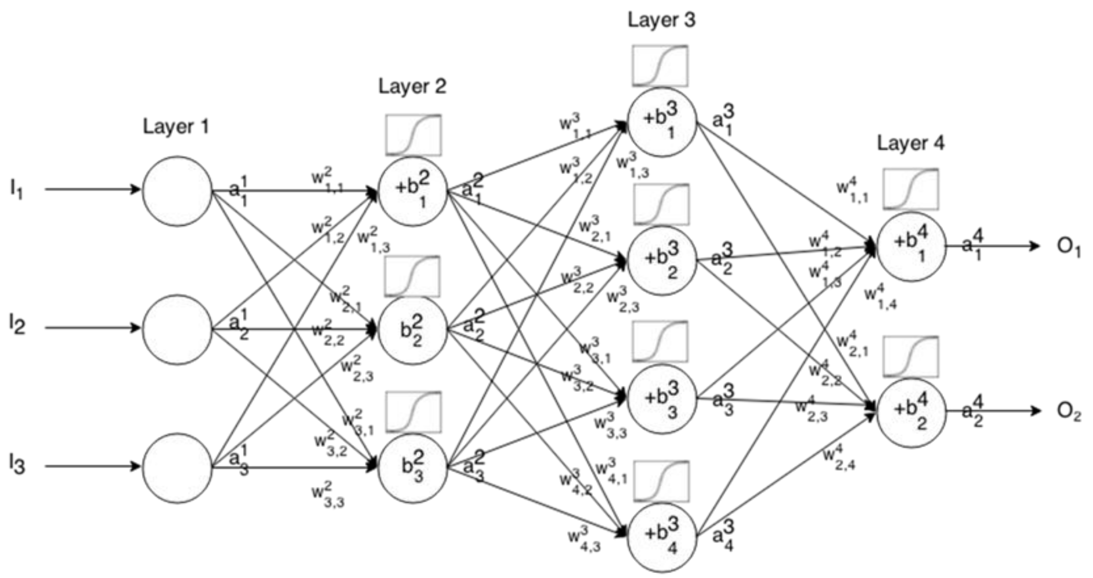




Neural Networks

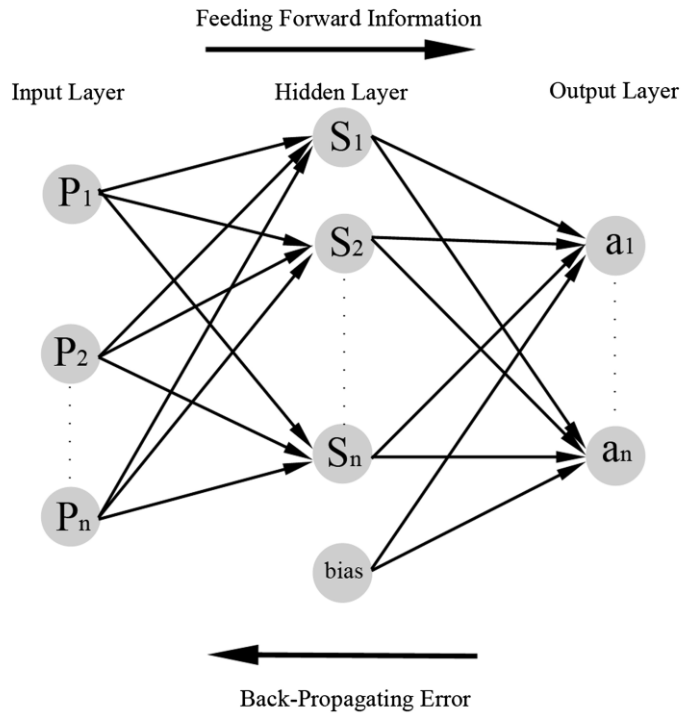
(*) A Neural Network (NN) consists of a network of McCulloch/Pitts computational neurons (a single layer was known historically as a "perceptron.")

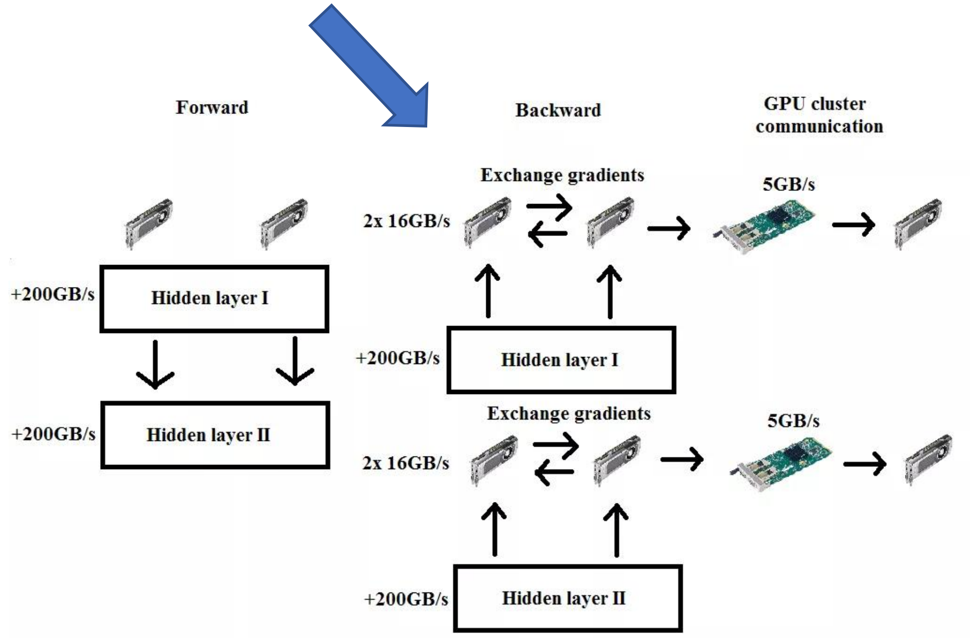
(*) NNs are *universal function approximators* – meaning that they can learn any arbitrarily complex mapping between inputs and outputs. While this fact speaks to the broad utility of these models, NNs are nevertheless prone to **overfitting**. The core issue in most ML/AI models can be reduced to the question of **generalizability**.



(*) Each neuron receives some inputs, performs a dot product and optionally follow it with a non-linearity (*e.g.* sigmoid/tanh). NNs are typically trained using *backpropagation*. This method calculates the gradient of a loss function (*e.g.* squared-loss) with respect to all the weights (W) in the network. More specifically, we use the chain rule to compute the 'delta' for the weight updates (one can think of this delta as assigning a degree of 'blame' for misclassifications).

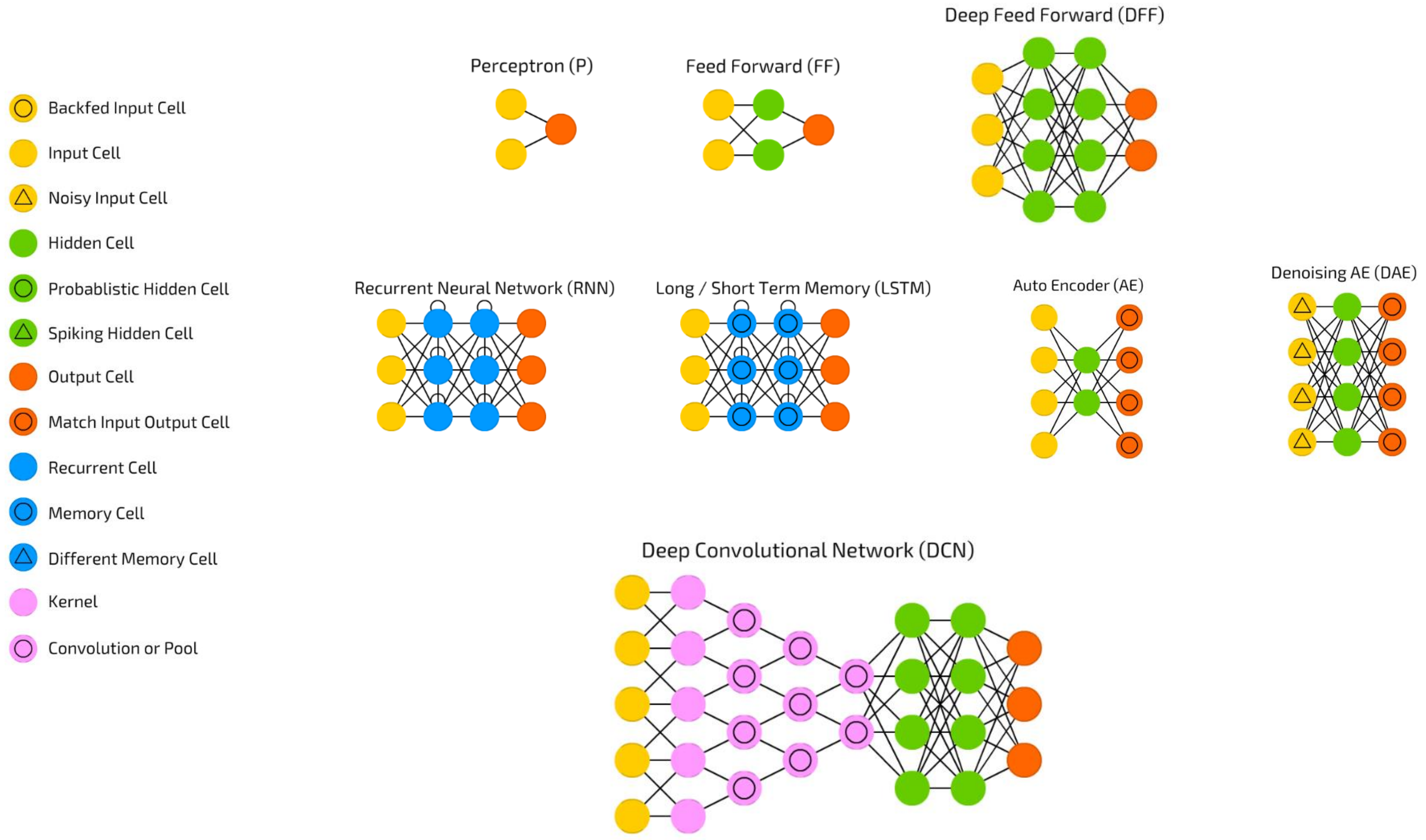
(*) Training a NN amounts to "tuning" the network weights. This process encompasses two distinct phases: (1) "**Forward phase**" in which case a datum is passed "forward" through the network (operationally this consists of a sequence of dot products and activations); (2) "**Backward phase**" during which the weights in the network are updated according to the desired output (i.e. label) for the datum; this process is <u>highly parallelizable</u>.



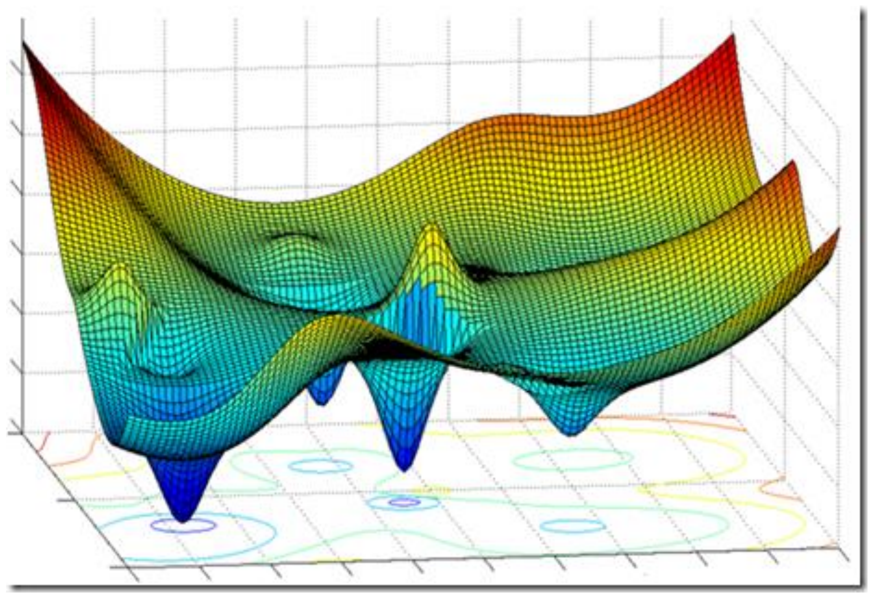


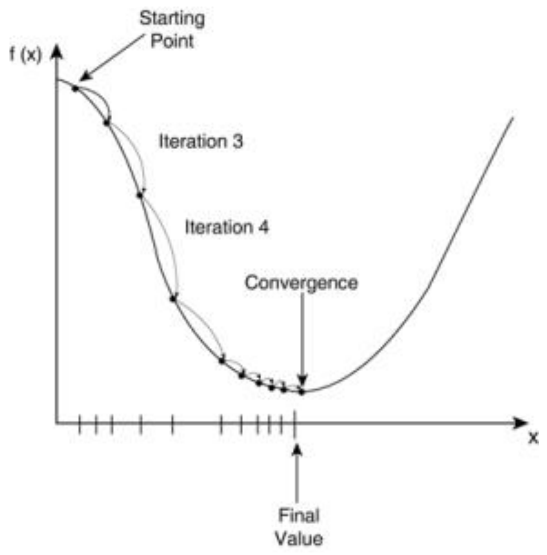
(*) Conventionally, NNs are best-suited for problems for which there exists a <u>large amount of (diverse and) labelled data</u>; training for deep learning (NNs with many layers/neurons) can be lengthy (recently training algorithms have become even more efficient); inference is however generally quick.

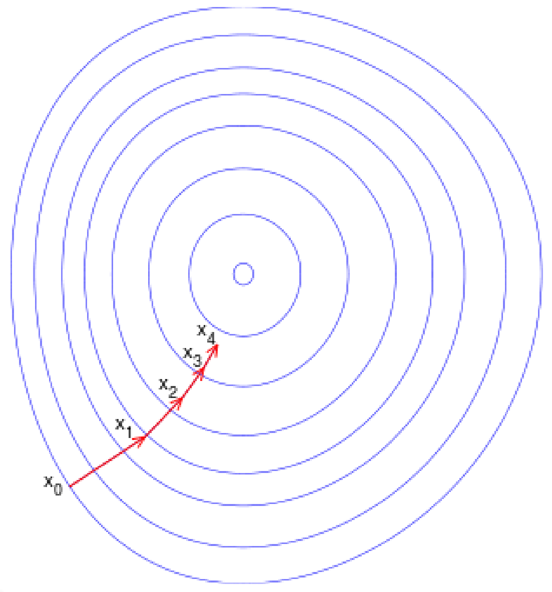
A Neural Network “Zoo”



Gradient Descent





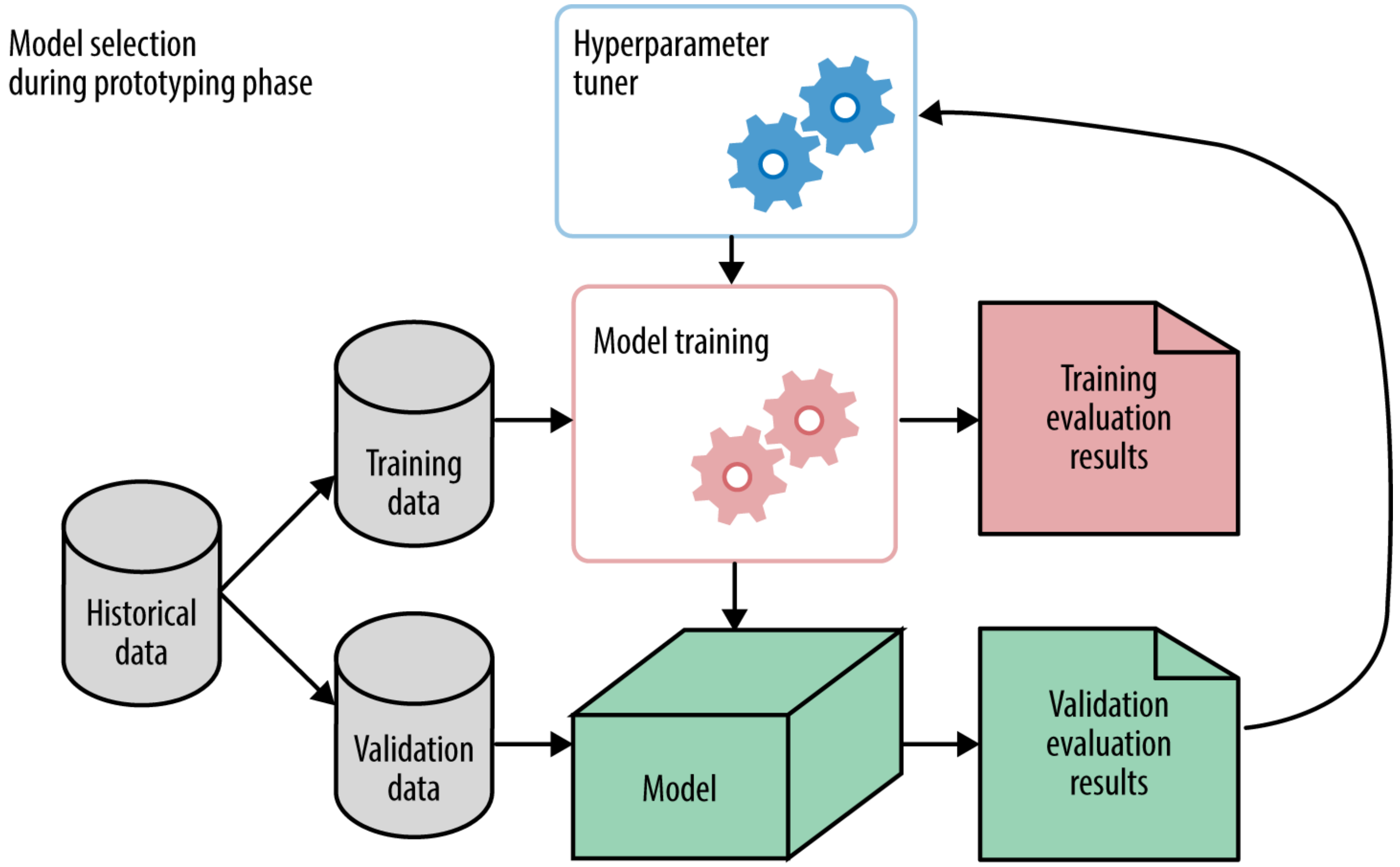


(*) Backpropagation is one particular instance of a larger paradigm of optimization algorithms know as **Gradient Descent** (also called "hill climbing").

(*) There exists a large array of nuanced methodologies for efficiently training NNs (particularly DNNs), including the use of **regularization**, **momentum**, **dropout**, **batch normalization**, pre-training regimes, initialization processes, etc.

(*) Traditionally, the backpropagation algorithm has been used to efficiently train a NN; more recently the **Adam stochastic optimization method** (2014) has eclipsed backpropagation in practice: https://arxiv.org/abs/1412.6980

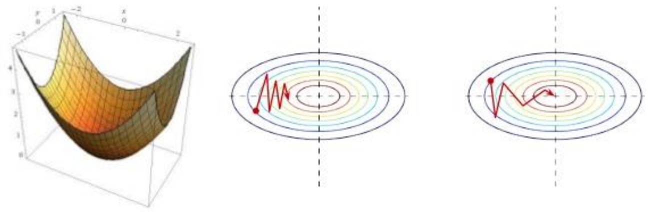
A Typical Workflow for Machine Learning



Some Deep Learning Techniques

- Momentum

Momentum



$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta \mathcal{L}(\theta_{t-1})$$
$$\theta_t = \theta_{t-1} - v_t$$

2x memory for parameters!

18

- L2 and L1 regularization

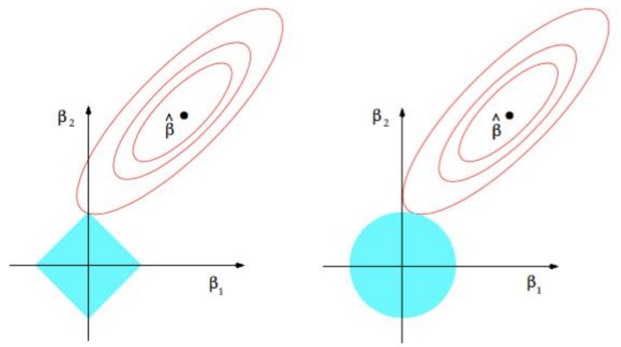


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \le t$ *and* $\beta_1^2 + \beta_2^2 \le t^2$, *respectively, while the red ellipses are the contours of the least squares error function.*

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

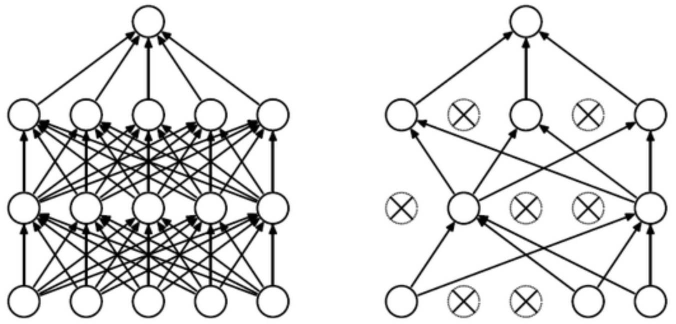
L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2}_{\text{Loss function}} + \lambda \underbrace{\sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

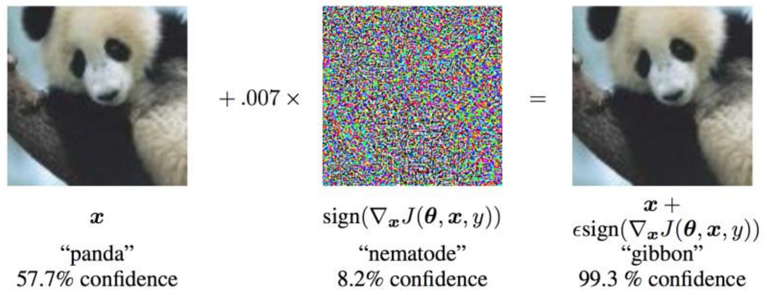
- Batch normalization

Some Deep Learning Techniques

- "Dropout" and sparse representations


(a) Standard Neural Net (b) After applying dropout.

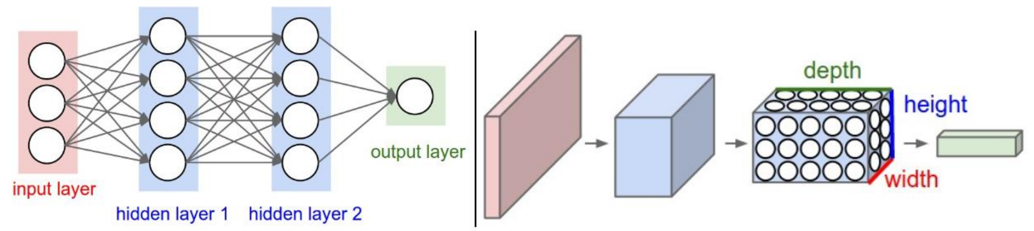
- Ensemble Modeling
- Parameter Initialization Strategies
- Adversarial Training


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

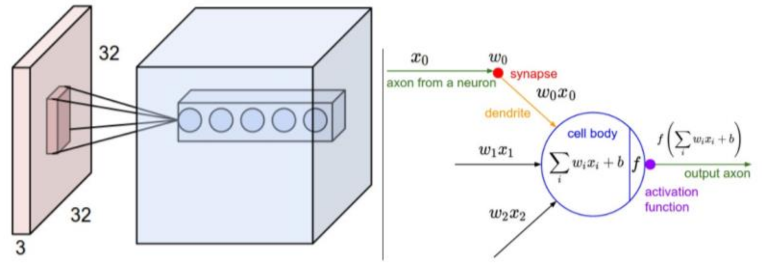
- Adam and Adaptive learning rate Deep Learning

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are very similar to ordinary NNs: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other.



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

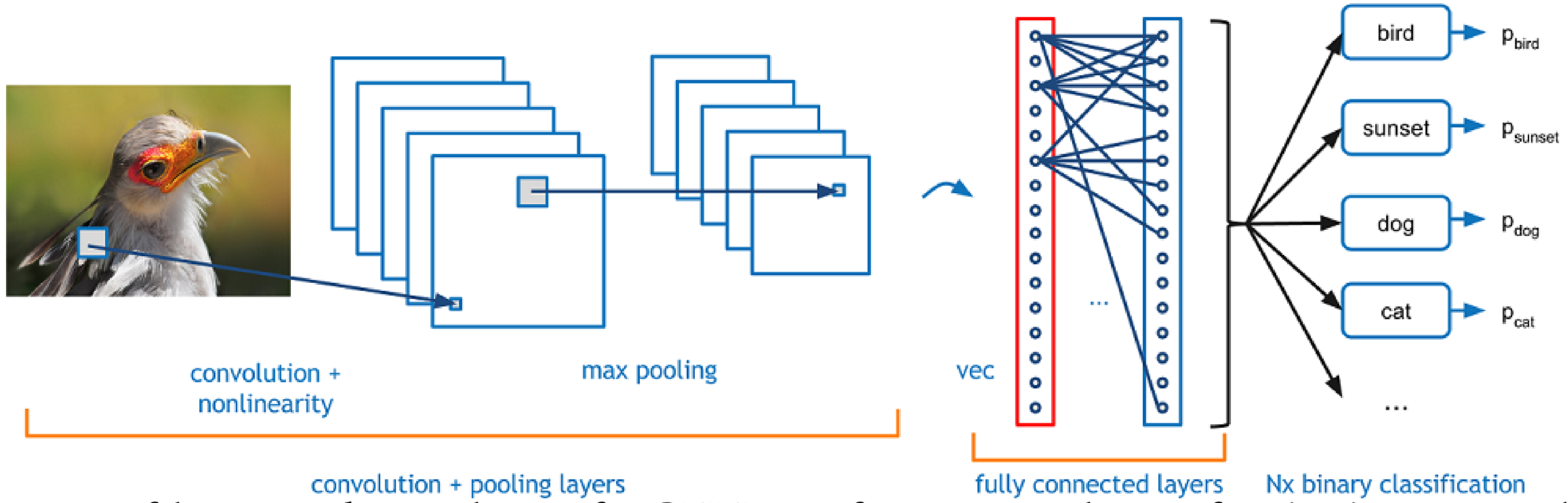


Left: An example input volume in red (e.g. a 32x32x3 CIFAR-10 image), and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels). Note, there are multiple neurons (5 in this example) along the depth, all looking at the same region in the input - see discussion of depth columns in text below. **Right:** The neurons from the Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.

The key difference with CNNs is that neurons/activations are represented as 3D volumes. CNNs additionally employ **weight-sharing** for computational efficiency; they are most commonly applied to image data, in which case image feature activations are trained to be translation-invariant (convolution + max pooling achieves this)

Convolutional Neural Networks

Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color. Now, we will have an entire set of filters in each CONV layer, and each of them will produce a separate 2-dimensional activations; these features are stacked along the depth dimension in the CNN and thus produce the output volume.



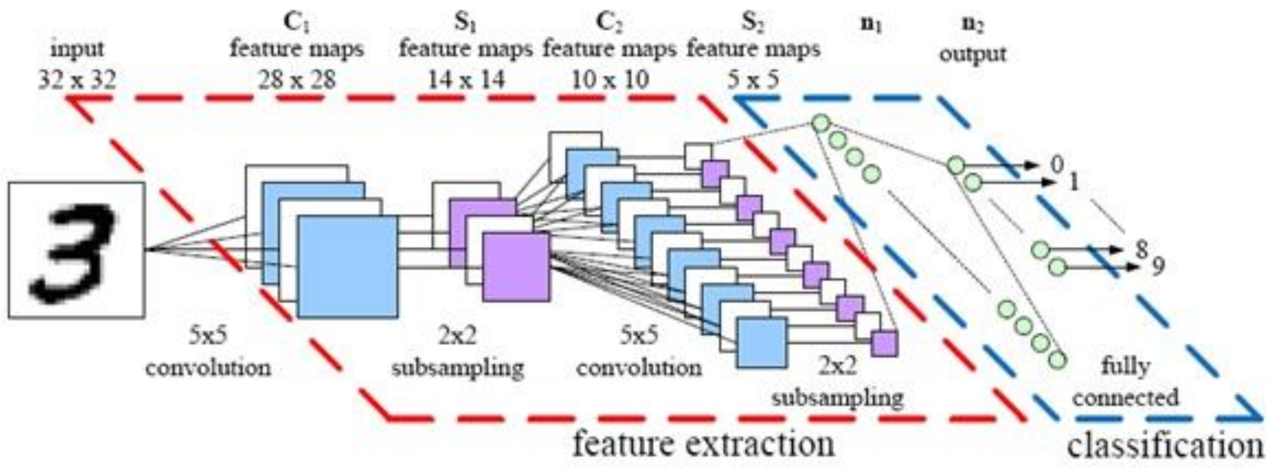
A simple CNN is a sequence of layers, and every layer of a CNN transforms one volume of activations to another through a differentiable function. The three main types of layers to build CNN architectures are: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer** (exactly as seen in regular Neural Networks). These layers are stacked to form a full CNN **architecture**.

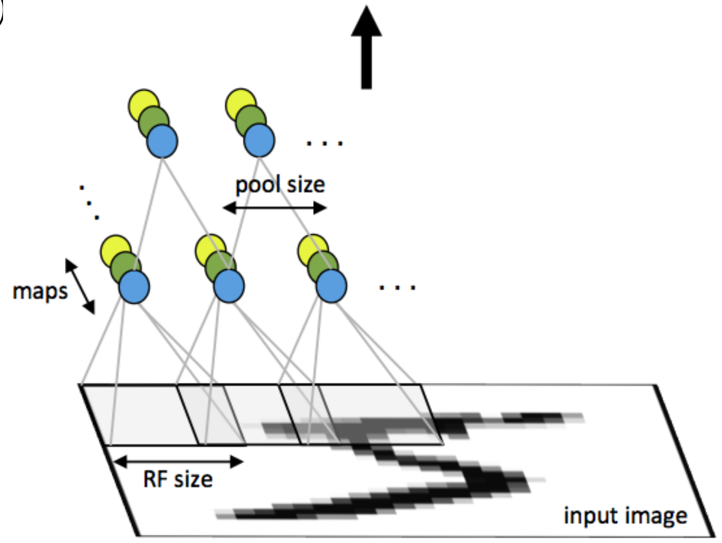
The convolution layer determines the activations of various filters over the original image; pooling is used for downsampling the images for computational savings; the fully-connected layers are used to compute class scores for classification tasks.

Convolutional Neural Networks

A nice way to interpret CNNs via a brain analogy is to consider each entry in the 3D output volume as an output of a neuron that looks at only a small region in the input and **shares parameters** with all neurons to the left and right spatially (since the same filter is used).

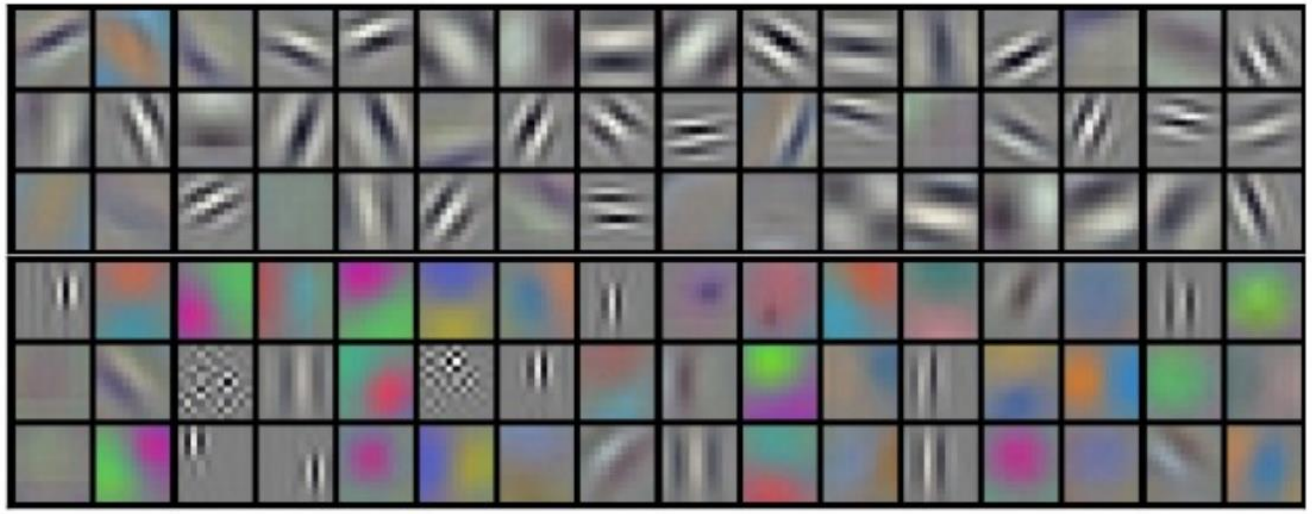
Each neuron is accordingly connected to only a local region of the input volume; the spatial extent of this connectivity is a hyperparameter called the receptive field (i.e. the filter size, such as: 5x5)





(Image from the LeCun MNIST paper, 1998)

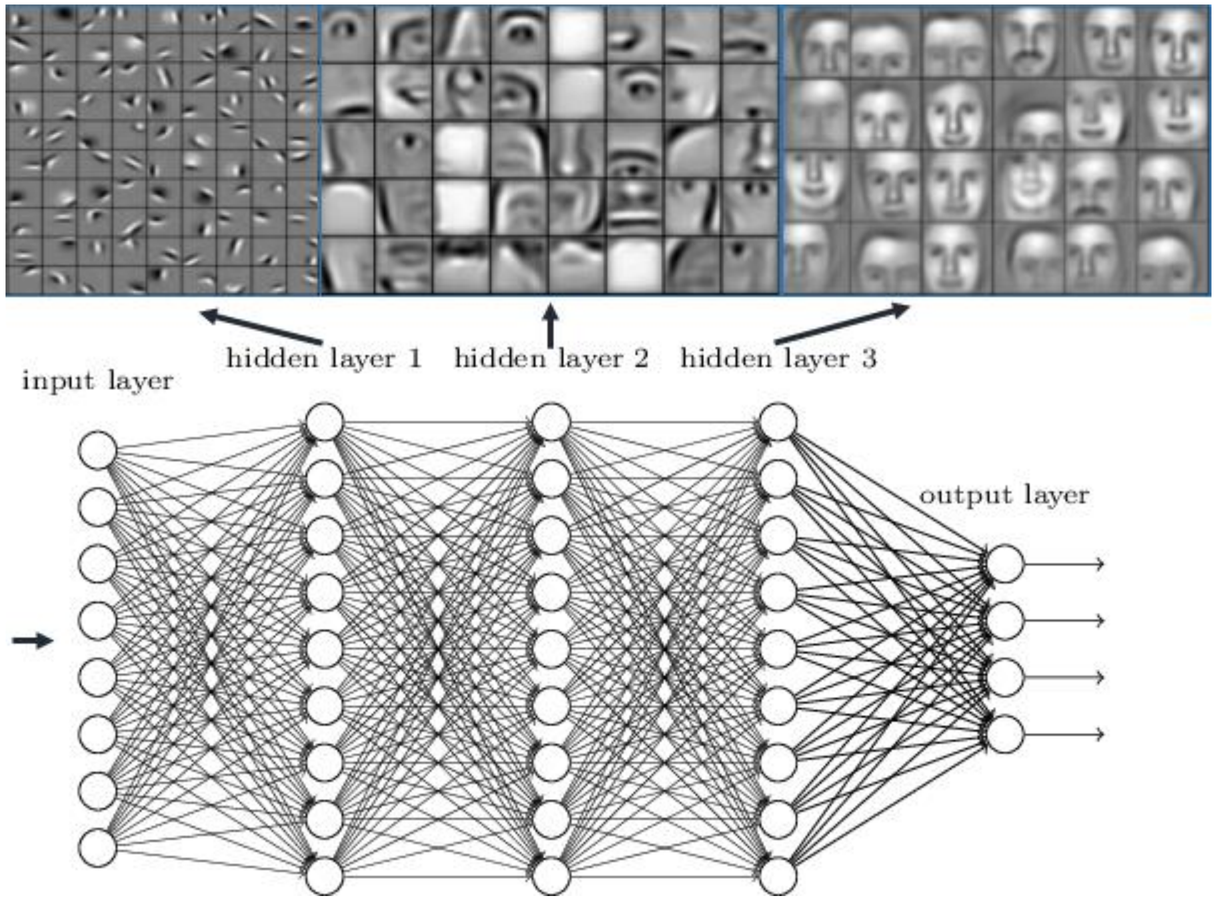
Convolutional Neural Networks

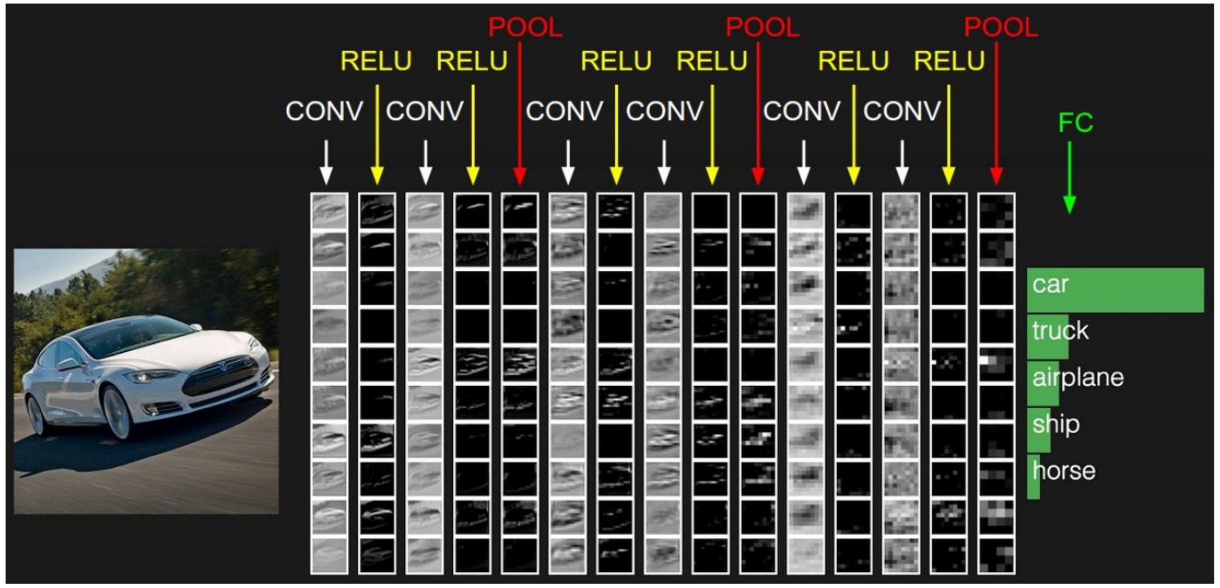


Example filters learned by Krizhevsky et al. Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the 55*55 neurons in one depth slice. Notice that the parameter sharing assumption is relatively reasonable: If detecting a horizontal edge is important at some location in the image, it should intuitively be useful at some other location as well due to the translationally-invariant structure of images. There is therefore no need to relearn to detect a horizontal edge at every one of the 55*55 distinct locations in the Conv layer output volume.

Of note, some researchers believe that the first stage of visual processing in the brain (called V1) serve as edge detectors that fire when an edge is present at a certain location and orientation in the visual receptive field.

DNNs Learn Hierarchical Feature Representations





A Very Brief History of Object Detection for Computer Vision

Viola-Jones Face Detection Algorithm

- P. Viola and M. J. Jones, *Robust real-time face detection*. International Journal of Computer Vision, 2004.

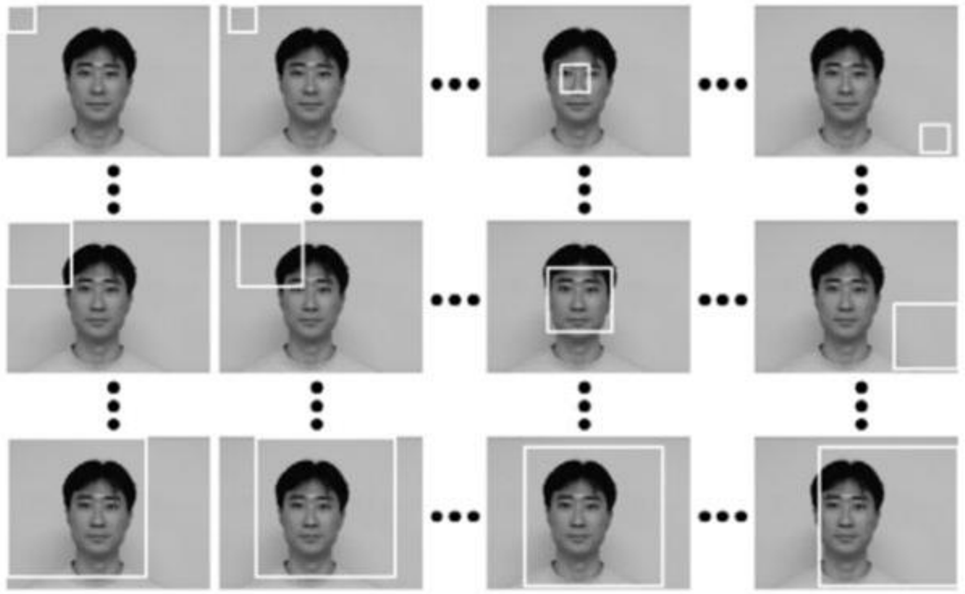
- First face-detection algorithm to work well in real-time (e.g., on digital cameras); it has been very influential in computer vision (17k+ citations).

Advantages

- Fast feature computation
- Efficient feature selection
- Scale and Location Invariant detector
- Instead of scaling image, scales features
- Can be trained for other types of objects

Disadvantages

- Detector is most effective only on frontal images of face
- Sensitive to lighting conditions
- Can generate multiple detections for the same face



Viola-Jones: Training Data

- Positive: Faces scaled and aligned to a base resolution of 24 by 24 pixels.

- Negative: Much larger number of non-faces.

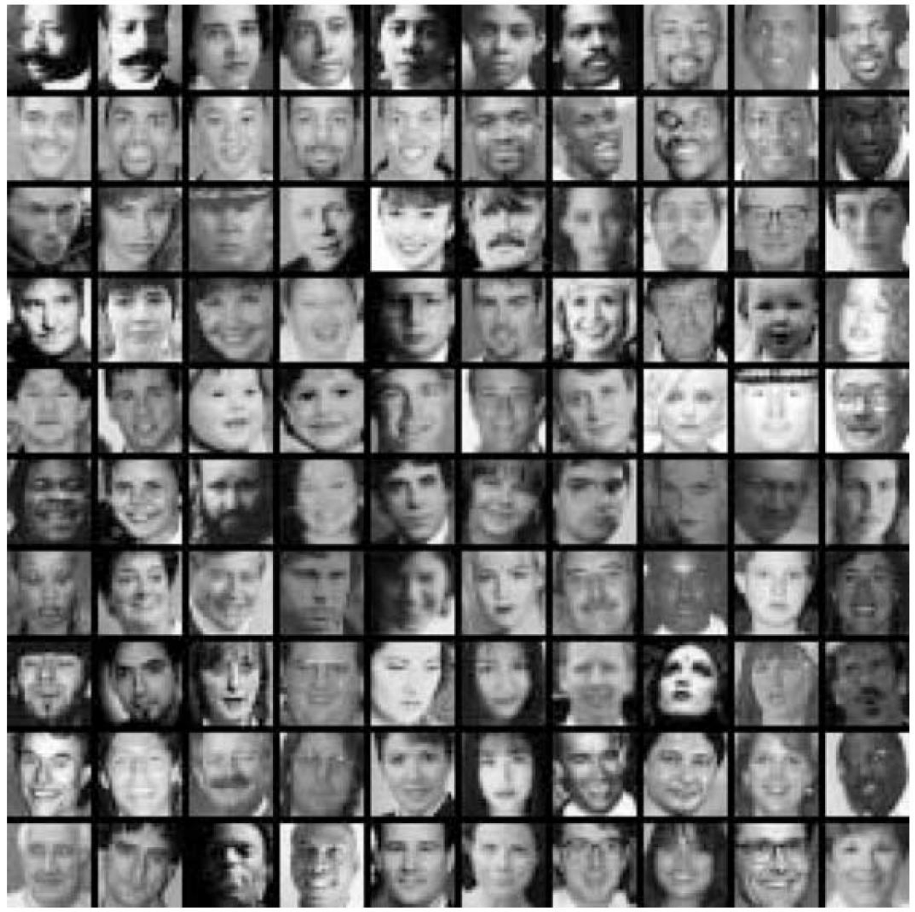
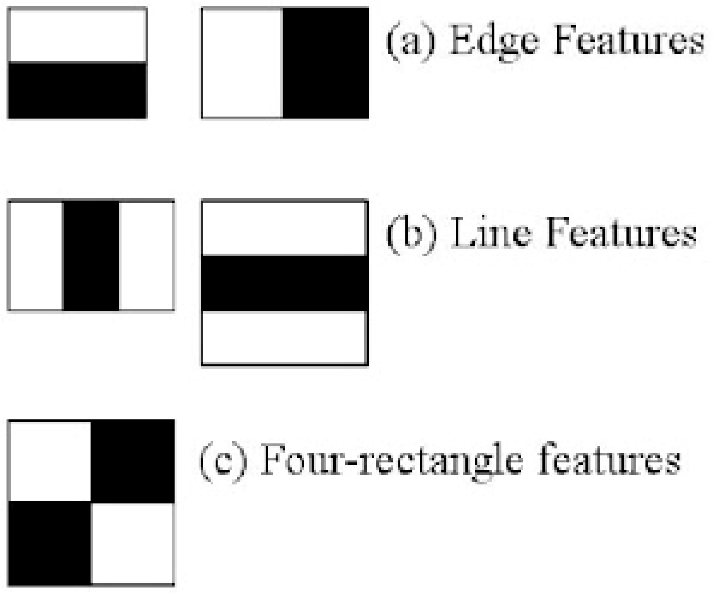
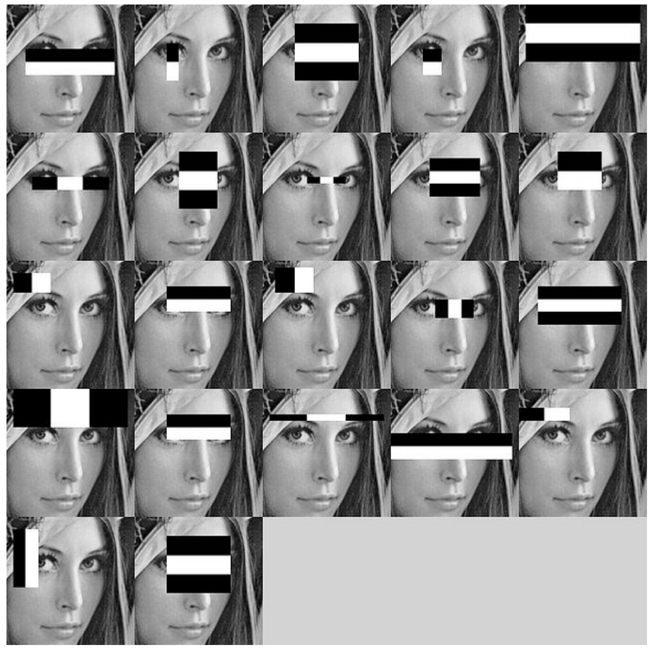


Figure 8. Example of frontal upright face images used for training.

Features





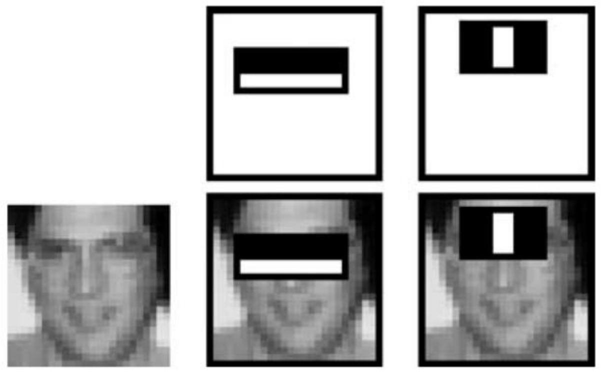


Figure 5. The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

Use rectangle features at multiple sizes and location in an image subwindow (candidate face). These features are known as Haar-like features.

For each feature f_j :

$$f_j = \sum_{b \in \text{black pixels}} \text{intensity(pixel } b) - \sum_{w \in \text{white pixels}} \text{intensity(pixel } w)$$

Possible number of features per 24 x 24 pixel subwindow > 180,000.

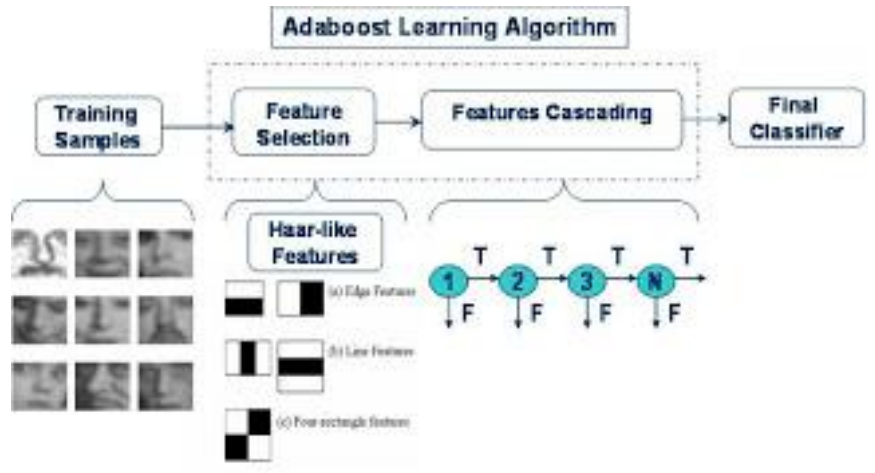
Detecting Faces

Given a new image:

- Scan image using subwindows at all locations and at different scales

- For each subwindow, compute features and send them to an ensemble classifier (learned via boosting). If classifier is positive ("face"), then detect a face at this location and scale.

- Algorithm uses a form of *AdaBoost* (popular boosting algorithm) to determine detection parameter for each feature/hypothesis, in addition to confidence measure for "fusion function" used for ensemble classification.



Histogram of Oriented Gradients (HOG features). 2005

(*) N. Dalal et al.

(*) Highly influential paper for CV (~22k citations)

(*) Main Idea: Local object appearance and shape can be described efficiently by the **distribution of intensity gradients/edge directions**.

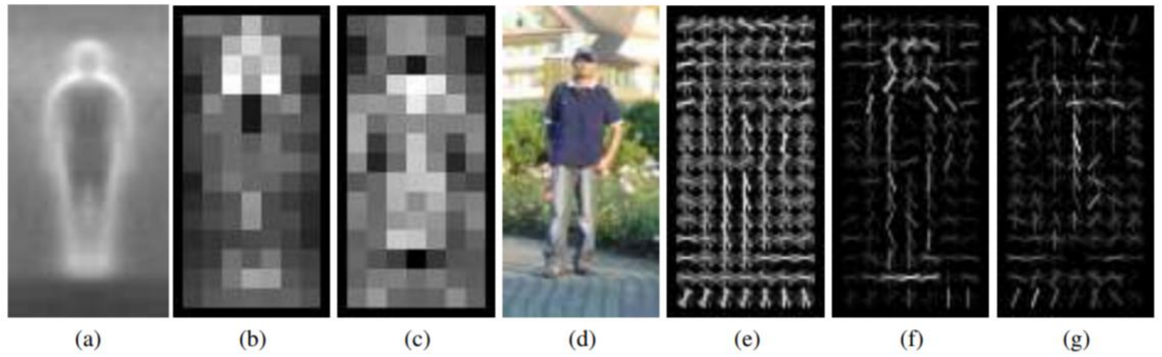
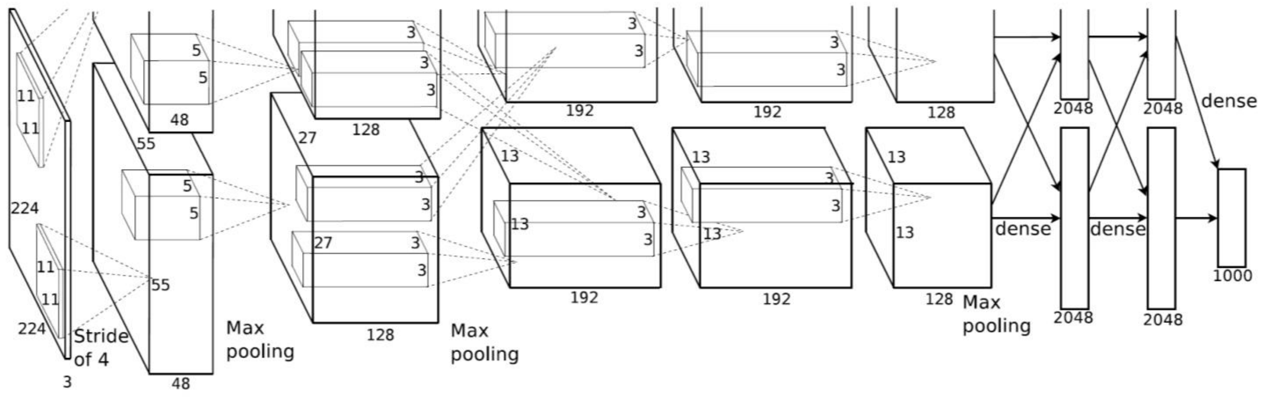


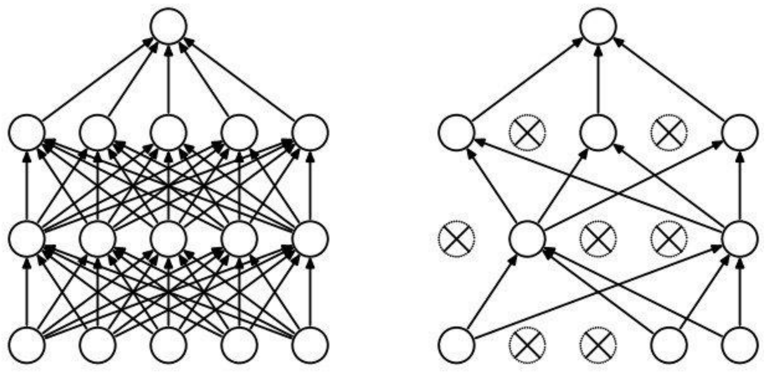
Figure 6. Our HOG detectors cue mainly on silhouette contours (especially the head, shoulders and feet). The most active blocks are centred on the image background just *outside* the contour. (a) The average gradient image over the training examples. (b) Each "pixel" shows the maximum positive SVM weight in the block centred on the pixel. (c) Likewise for the negative SVM weights. (d) A test image. (e) It's computed R-HOG descriptor. (f,g) The R-HOG descriptor weighted by respectively the positive and the negative SVM weights.

(*) The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing.

(*) HOG features advantage: <u>invariant to geometric and photometric transformations</u>; HOG features are particularly good at detection people.

DNNs: AlexNet (2012)





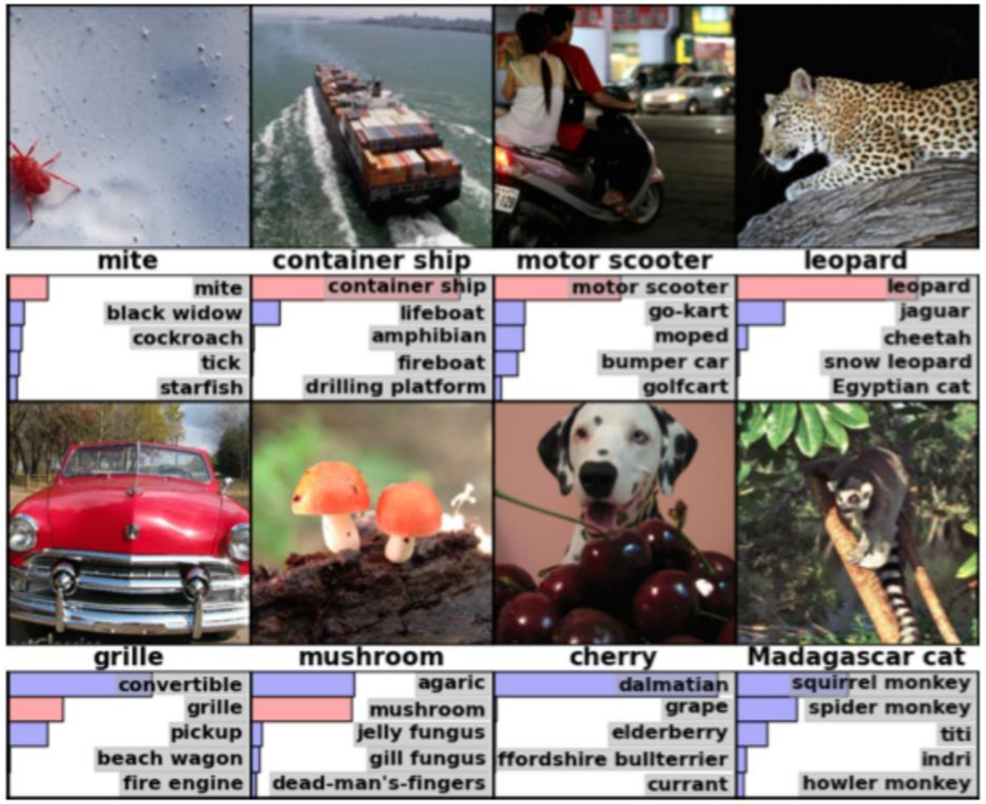
(a) Standard Neural Net (b) After applying dropout.

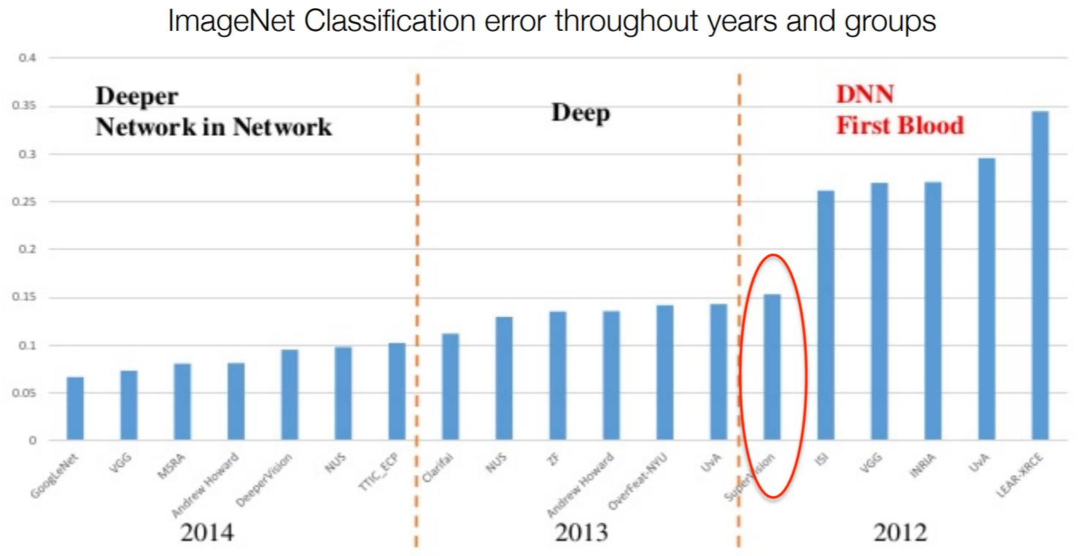
AlexNet was developed by Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever; it uses CNNs with GPU support. The network achieved a top-5 error of 15.3%, more than 10.8 percentage points ahead of the runner up.

Among other innovations: AlexNet used GPUs, utilized RELU (rectified linear units) for activations, and "dropout" for training.

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

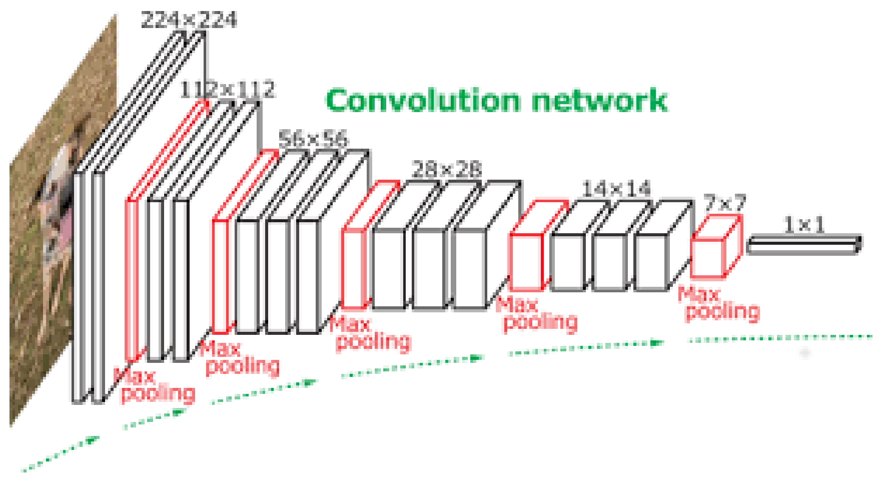
DNNs: AlexNet (2012)





Li Fei-Fei: ImageNet Large Scale Visual Recognition Challenge, 2014 http://image-net.org/

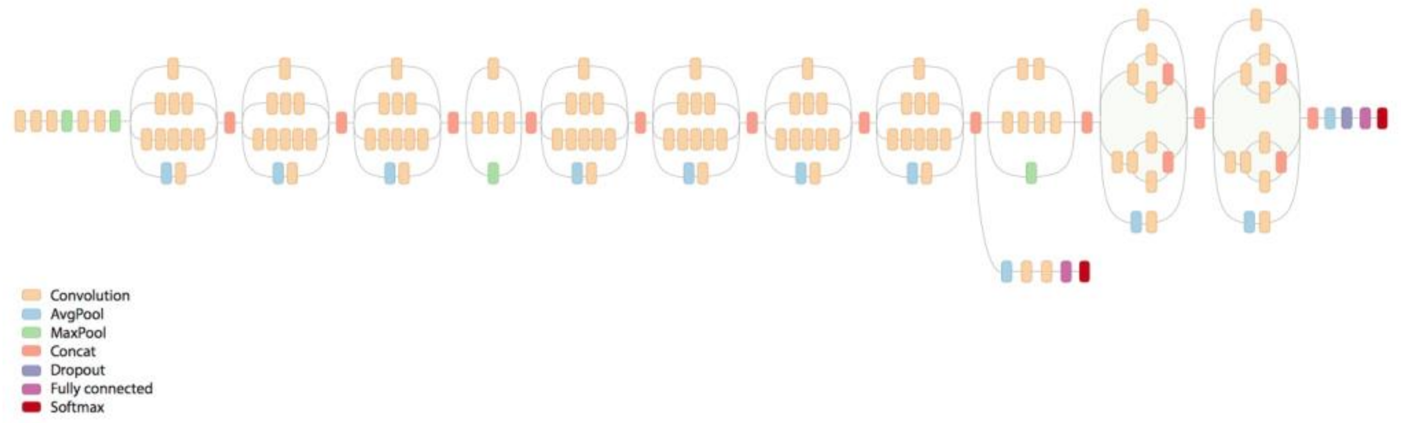
DNNs: VGG (2014)



- Team at Oxford produced influential DNN architecture (VGG).Using very small convolutional filters (3x3), they achieved a significant improvement on the prior-art configurations by <u>pushing the depth to 16–19 weight layers</u>.
- Team achieved first and second place on the ImageNet Challenge 2015 for both localization and classification tasks, respectively.
- Using pre-trained VGG is very common practice in research.

https://arxiv.org/pdf/1409.1556.pdf

DNNs: Inception (2015, Google)

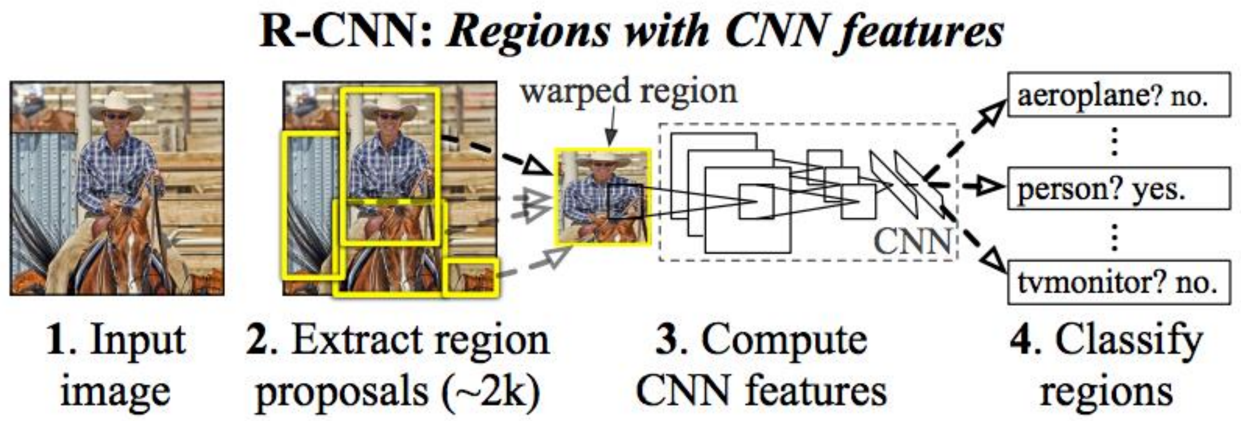


https://arxiv.org/pdf/1409.1556.pdf

- Team at Google (Szegedy et al.) produced an even deeper DNN (22 layers). No need to pick filter sizes explicitly, as network learns combinations of filter sizes/pooling steps; upside: newfound flexibility for architecture design (architecture parameters themselves can be learned); downside: ostensibly requires a large amount of computation – this can be reduced by using 1x1 convolutions for dimensionality reduction (prior to expensive convolutional operations).
- Team achieved new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14; 6% top-5 error rate for classification.

Object Detection with Deep Learning: RCNN

- Girshick et al.*achieved state-of-the-art performance on several object detection benchmarks using a "regions with convolutional neural networks" (R-CNN).



- R-CNN and its extensions use a *region-proposal network* (RPN) that simultaneously predicts object bounds and *objectness* scores for proposals.
- To avoid an exhaustive search, R-CNN utilizes a *selective search algorithm* that reduces the overall computational overhead requirement.

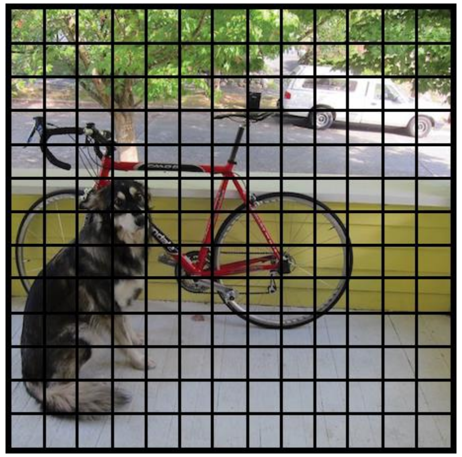
*R. Girshick, Fast R-CNN, in: Int. Conf. Comput. Vis., IEEE, 2015: pp. 1440–1448;

*R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Conf. Comput. Vis. Pattern Recognit., IEEE, 2014: pp. 580–587

Object Detection with Deep Learning: YOLO

- YOLO only looks at the input image once (hence the name).

- First the algorithm divides the input image into a grid of 13x13 cells.



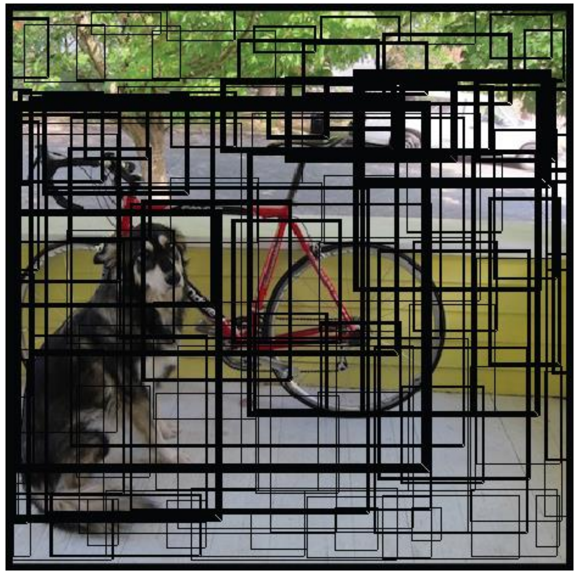
- Each of these cells is responsible for predicting 5 bounding boxes. A bounding box describes the rectangle that encloses an object.

- YOLO also outputs a *confidence score* that tells us how certain it is that the predicted bounding box actually encloses some object. This score doesn't say anything about what kind of object is in the box, just if the shape of the box is any good.

Object Detection with Deep Learning: YOLO

- The predicted bounding boxes may look something like the following (the higher the confidence score, the fatter the box is drawn):

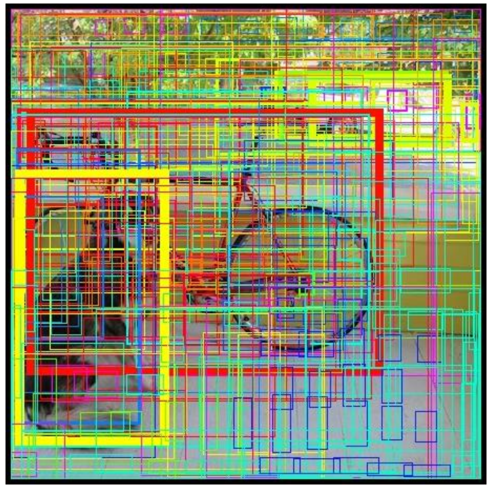
- For each bounding box, the cell also predicts a *class*. This works just like a classifier: it gives a probability distribution over all the possible classes.

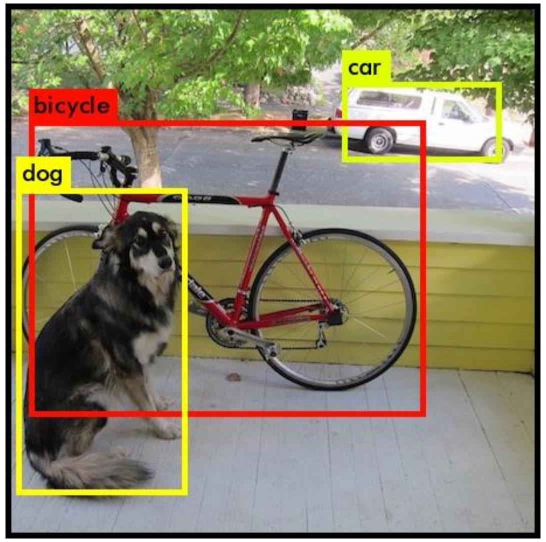


https://arxiv.org/pdf/1506.02640.pdf

Object Detection with Deep Learning: YOLO

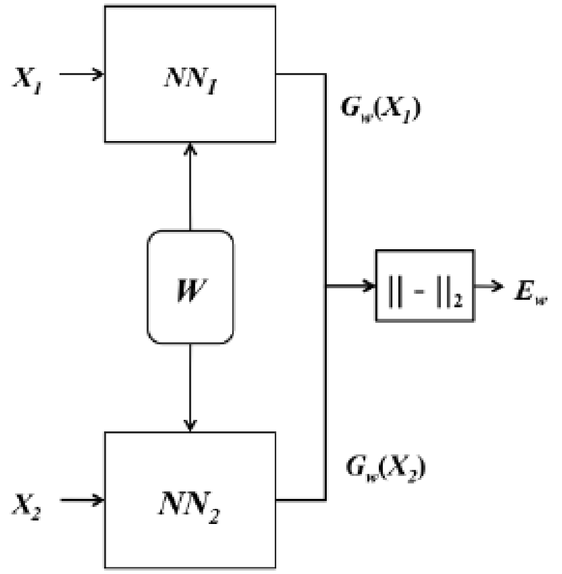
- The confidence score for the bounding box and the class prediction are combined into one final score that tells us the probability that this bounding box contains a specific type of object. For example, the big fat yellow box on the left is 85% sure it contains the object “dog”:





- Since there are 13×13 = 169 grid cells and each cell predicts 5 bounding boxes, we end up with 845 bounding boxes in total. It turns out that most of these boxes will have very low confidence scores, so we only keep the boxes whose final score is 30% or more (you can change this threshold depending on how accurate you want the detector to be).

Similarity Learning with DNNs



- Similarity learning is the process of training a metric to compute the similarity between two entities (also called: *metric learning*).

- Example applications: Face recognition, customer identification, visual search/recognition of products in store.

- A Siamese Network is a NN where the network is trained to distinguish between two inputs.

- Typically a Siamese network uses two encoders (the "sister" networks – they can be identical); each encoder is fed with one of the images in either a positive or negative pair.

- The network is trained using "contrastive loss" – the Euclidean distance of the image encodings. Optimization is performed using a standard algorithm, e.g., backprop, Adam.

Similarity Learning with DNNs

- Importantly: Siamese networks can be used as **one shot classification** models, on the other hand, which require just **one training example** of each class you want to predict on.

- A nice example would be facial recognition. You would train a One Shot classification model on a dataset that contains various angles , lighting , etc. of a few people. Then if you want to recognize if a person X is in an image, you take one single photo of that person, and then ask the model if that person is in the that image(*note, the model was not trained using any pictures of person X*).

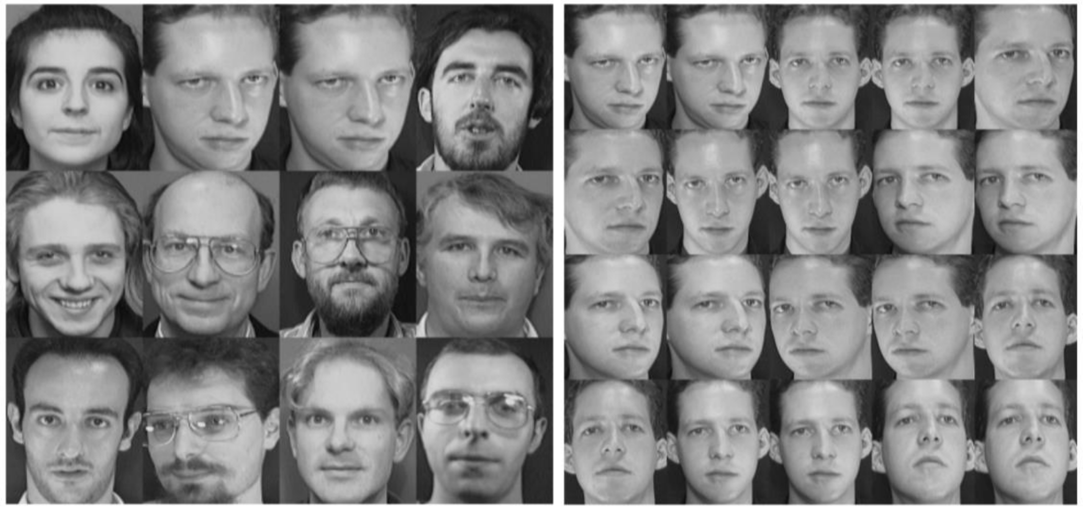


Figure 1.0. **Left**: Samples from different classes. **Right**: All Samples of one subject

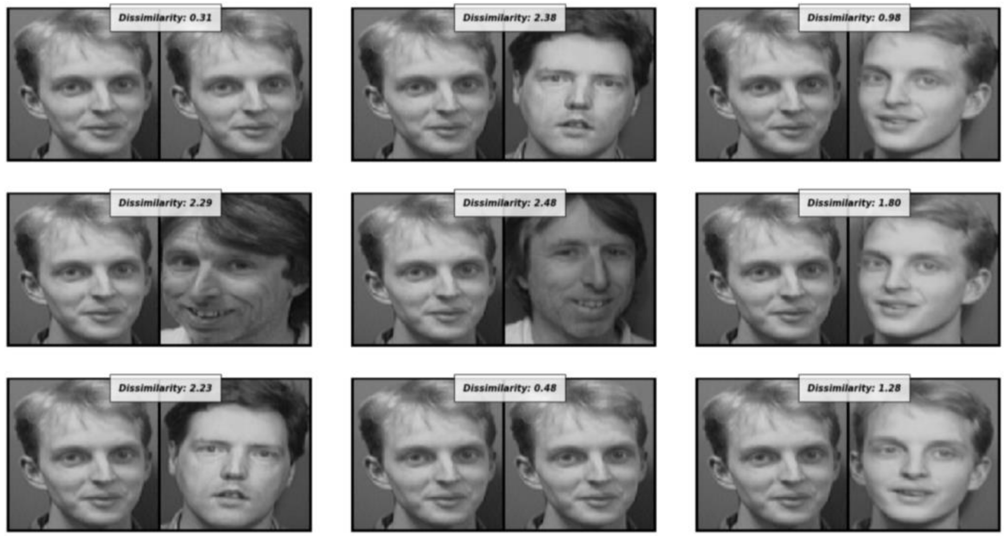
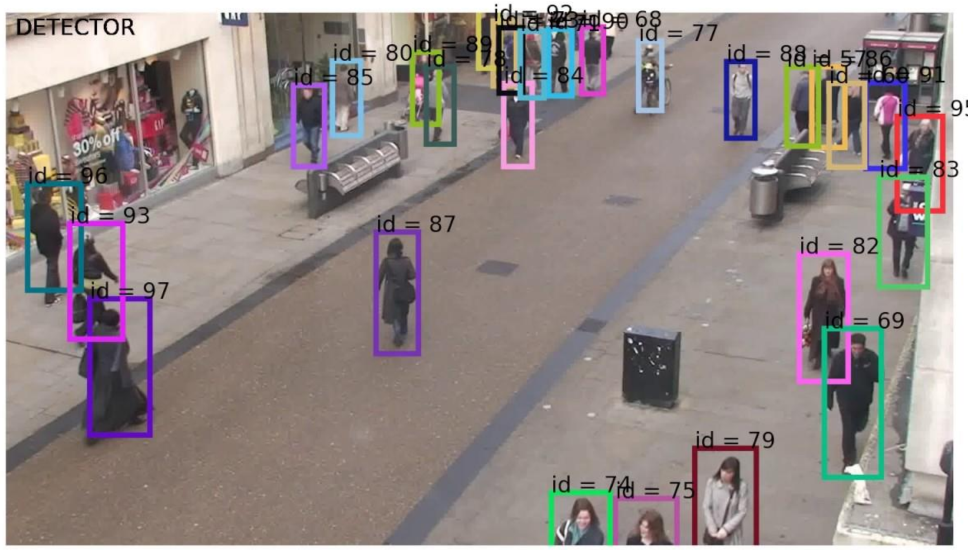


Figure 3.0 Some outputs of the model. Lower values indicate more similarity, and higher values indicate less similarity.

Multiple-Object Tracking (MOT)

- The task of *Multiple Object Tracking* (MOT) is generally divided between locating multiple objects, maintaining their identities and generating their individual trajectories.

- As a mid-level task in CV, MOT can serve as an intermediate step for tasks such as pose estimation, action recognition and behavior analysis.



Multiple-Object Tracking (MOT)

- Compared to single object tracking (SOT), MOT requires two additional tasks to be solved: (1) determining the number of objects, and (2) maintaining their identities.

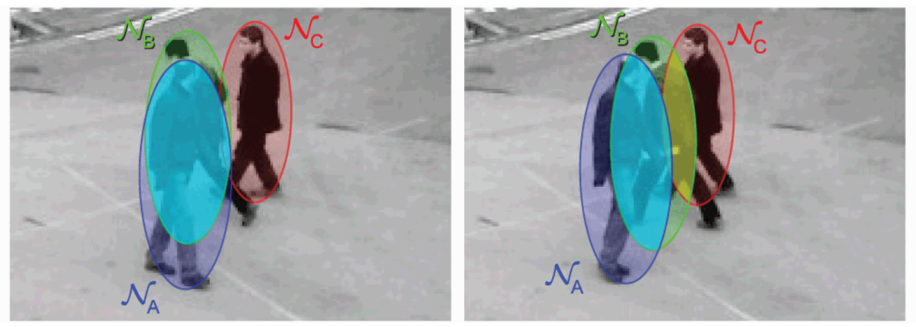
- Apart from the common challenges shared by SOT and MOT, further key issues that complicate MOT include among others:

  (i) frequent occlusion

  (ii) initialization and termination of tracks

  (iii) similar appearance of objects

  (iv) Interactions amongst multiple objects



Multiple-Object Tracking (MOT)

- Formulation of problem:

- MOT is a multi-variate estimation problem.

- Given an image sequence, denote the state of the *ith* object in the *t-th* frame as s_t^i; accordingly, let $S_t = (s_t^i, \dots, s_t^{M_t})$ connote the states of all M_t objects in the *t-th* frame: $S_{1:t} = \{S_1, \dots, S_t\}$ represents all sequential states of all objects from the first frame to the *t-th* frame.

- Similarly, define $O_t = (o_t^i, \dots, o_t^{M_t})$ as the collected observations for all of the M_t objects in the *t-th* frame; $O_{1:t}$ is defined analogously.

Multiple-Object Tracking (MOT)

- With these notational conventions, the objective of MOT is to find the optimal sequential states of all objects, which can generally be modeled by performing MAP estimation from the conditional distribution of the sequential states give all the observations:

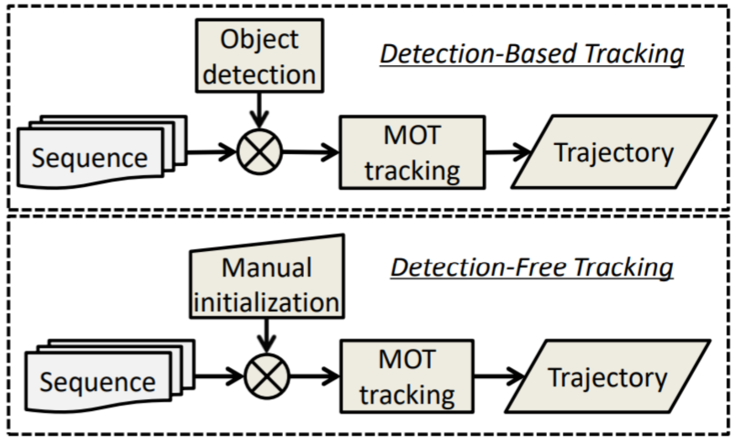
$$\hat{S}_{1:t} = \arg\max_{S_{1:t}} P\left(S_{1:t} \mid O_{1:t}\right)$$

- The probabilistic inference based approach usually solves the MAP estimation using a two-step iterative procedure:

(1) Prediction: $P(S_t | O_{1:t-1})$

(2) Update: $P(S_t | O_{1:t}) = P(O_t|S_t)P(St|O_{1:t-1})$

Multiple-Object Tracking (MOT)



- Most MOT frameworks can be grouped into (2) sets:
- (1) Detection-based Tracking (DBT)
- (2) Detection-free Tracking (DFT)

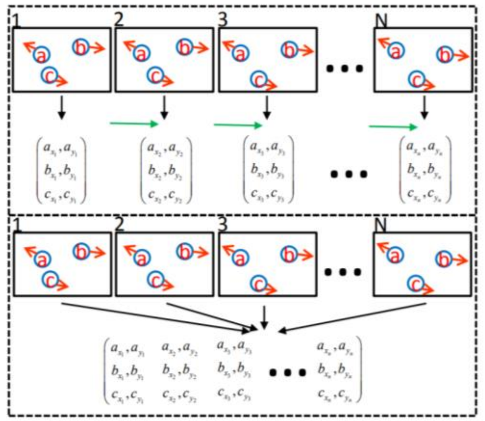
- DBT: Objects are first detected and then linked into trajectories. Two key issues for DBT framework: object detector must be trained in advance (offline); performance of tracking algorithm is highly dependent upon effectiveness of detector (e.g. RCNN, YOLO).

- DFT: Requires manual initialization of fixed number of objects in first frame, then localizes objects in subsequent frames.

- DBT is more popular because new objects discovered and disappearing objects are terminated automatically. DFT can't accommodate cases for which objects appear – but it is nevertheless free of pre-trained detectors (no deep learning needed).

Multiple-Object Tracking (MOT)

- Processing Modes for MOT: online tracking or offline tracking.
- Online, tracking methods only rely on the past information available up to the current frame, while offline approaches employ observations from both past and future.
- For real-time tracking we require online tracking – but this doesn't rule out post-processing step (offline) if say, system presents low confidence for tracking/gesture recognition, etc. (This would require temporarily saving some video streams which might be problematic).



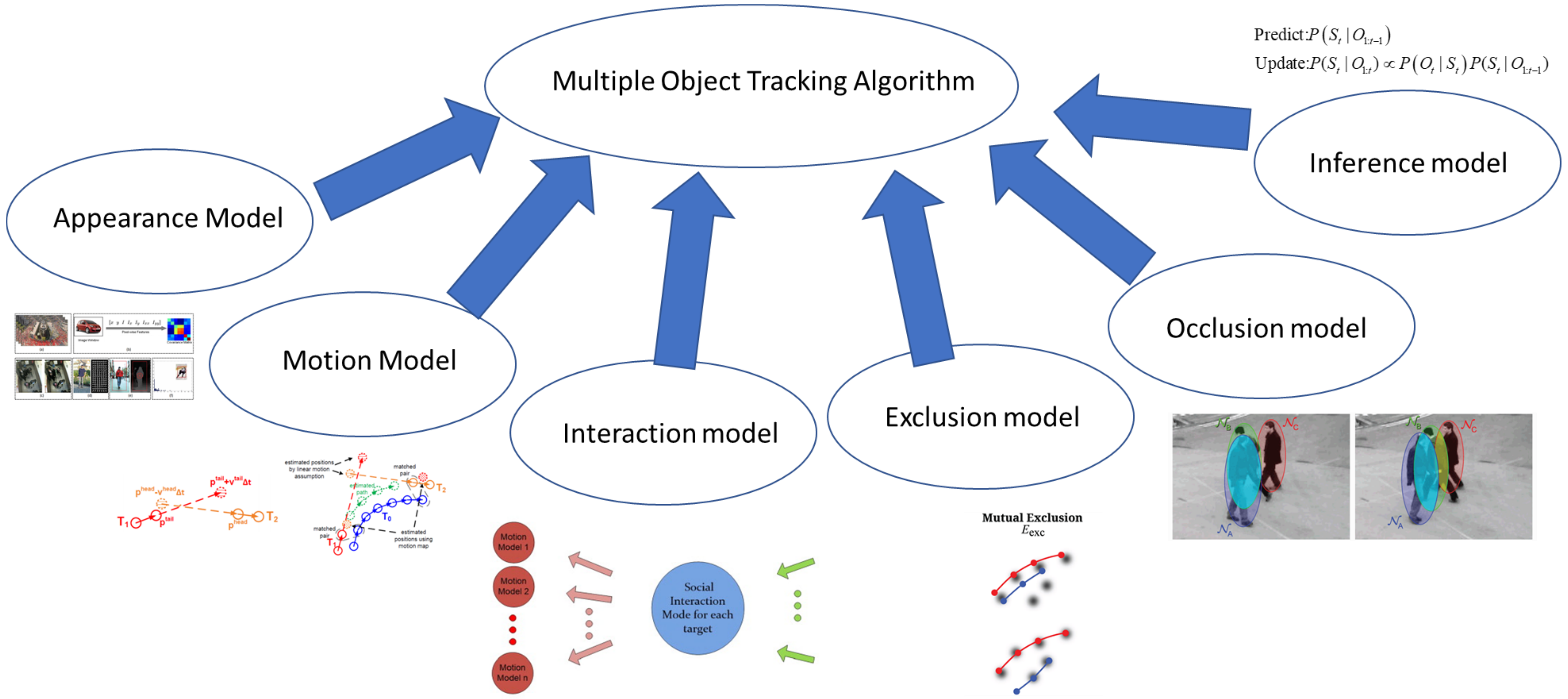
Multiple-Object Tracking (MOT)

- GOAL: discover multiple objects (simultaneously) in individual frames and recover the identity information across continuous frames (i.e. trajectory)

- (2) major considerations: (1) how to measure similarity (if at all) between objects in frames

- (2) how to recover identity information based on similarity measure across frames (inference problem)

Components of an MOT Algorithm

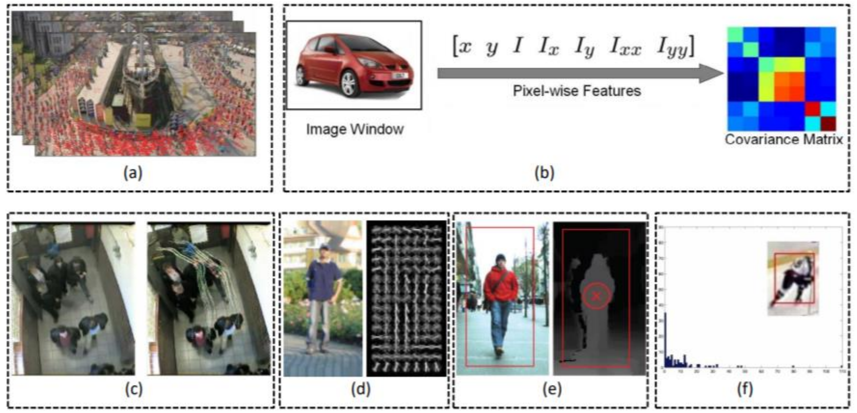


Predict: $P(S_t \mid O_{1:t-1})$

Update: $P(S_t \mid O_{1:t}) \propto P(O_t \mid S_t) P(S_t \mid O_{1:t-1})$

Multiple-Object Tracking (MOT): Components of MOT

- (A) Appearance Model

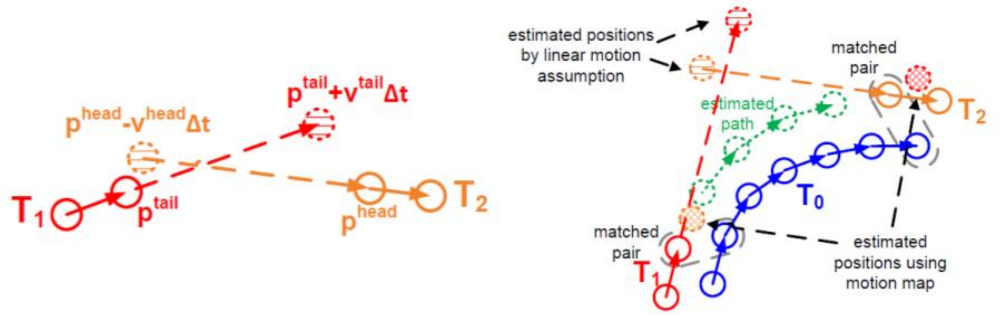


- Appearance is an important cue for affinity computation in MOT. However, it is usually not considered the core component of a MOT algorithm.

- Appearance models consist of (i) visual representation (e.g. local features or region features such as optical flow, HOG features, region covariance) and (ii) statistical measuring (using single cues, e.g., distance between color histograms or multiple cues involving a fusion of information, e.g. boosting, cascading, etc.)

Multiple-Object Tracking (MOT): Components of MOT

- (B) Motion Model



$p^{head} - v^{head}\Delta t$
$p^{tail} + v^{tail}\Delta t$
T_1
p^{tail}
p^{head}
T_2
estimated positions by linear motion assumption
matched pair
estimated path
T_2
T_0
matched pair
T_1
estimated positions using motion map

- The motion model captures the dynamic behavior of an object. In most cases objects are assumed to move smoothly and thereby maintain object continuity conditions.

- (i) Linear Motion Model: the most popular motion model for MOT, assumes constant velocity of objects; one can further impose velocity/position/acceleration smoothness constrains.

- (ii) Non-linear motion model: more complex model (often) can yield better results when linking tracklets of objects.

Multiple-Object Tracking (MOT): Components of MOT

- (C) Interaction Model

- Captures influence of an object on other objects. Example: object experiences "force" from other objects, i.e. customer adjusts speed/trajectory to avoid others.

- Social forces model: target behavior is modeled based on two aspects: individual force and group force; use continuity and fidelity constraints (smoothness and desired destination of subject doesn't change); group force: use attraction (individuals moving together stay together), repulsion (groups maintain some distance from one another), coherence (group elements move with commensurate velocity, etc.)

Multiple-Object Tracking (MOT): Components of MOT

- (D) Exclusion Model

- Constraint set employed to avoid physical collisions for MOT problem; intuition: two objects can't occupy the same physical space. Two types: (1) detection-level exclusion (can't assign detection to same space in same frame for two distinct customers), (2) trajectory-level exclusion (trajectories must maintain some minimal epsilon difference).

- For in-store environment, exclusion model could incorporate intrinsic exclusions based on surrounding environment.

- To model exclusion: can use exclusion graph to capture constrain; optimize inference to encourage connected nodes to have different labels; further distinction between soft/hard modeling.

Multiple-Object Tracking (MOT): Components of MOT

- (E)Occlusion Handling (critical challenge):

- Occlusion is a common cause of ID switches and fragmentation of trajectories.

Remedies:

- (i) Part-based model: divide holistic object into several parts and compute affinity based on individual parts; in case of occlusion, only non-occluded parts contribute to affinity measure; (ii) buffer strategy: buffer observations when occlusion occurs, when occlusion ends, object states are recovered based on the buffered observations and the stored states before occlusion.

Multiple-Object Tracking (MOT): Components of MOT

- (F) Inference

- Probabilistic inference: Usually relies on a Markov assumption: (i) current object state only depends on previous state; (ii) observation of an object is only relation to its state corresponding to this observation (observations are conditionally independent). States of objects can be estimated by iteratively conducting the prediction and updating steps.

- One solution: Kalman fitter (could use Gaussian Process): makes linear system assumption and Gaussian-distributed object states assumption; under these conditions, Kalman filter is optimal estimator.

Multiple-Object Tracking (MOT): Components of MOT

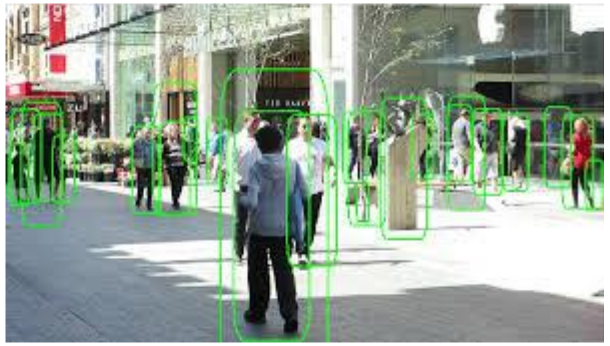
- (F) Inference

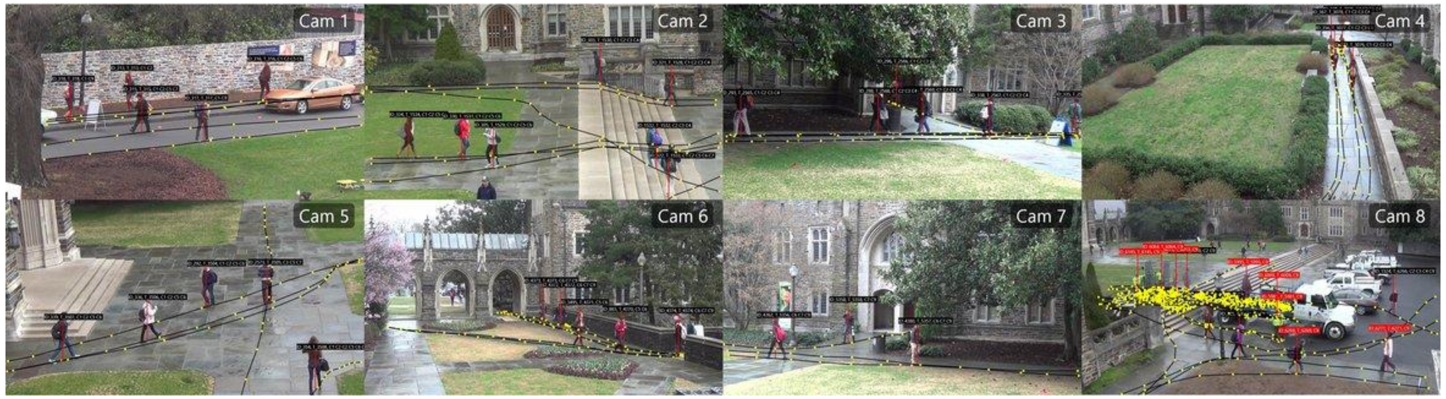
- Can also model MOT problem as graph bipartite matching problem; two sets of nodes: observations/detections; weights between nodes are affinity measure; can solve greedily or use Hungarian algorithm (classic paradigm for matching with preference problem: "stable marriage problem"; can alternatively use min-cut/max flow network (use for trajectories); graphical models/conditional random fields.

Multiple-Object Tracking (MOT)

MOT evaluation: standard metrics: accuracy/precision; tracking metrics: MOTA metric (mot accuracy) combines FP rate and FN rate and mismatch rate; precision: uses IOU metric and/or distance; completeness: how completely the ground truth trajectories were tracked; robustness: recovery from occlusion measure.

Datasets: MOTChallenge, KITTI, DukeMTMCT





Open source: (surprisingly few for MOT): more for SOT; RCNN, Fast RCNN, Faster RCNN, YOLO, MOSSE Tracker, SORT, DEEPSORT, INTEL SDK OPENCV.

Multiple-Object Tracking (MOT)

- Issues: DBT methods are highly dependent on classifier (good news – these are improving every few months); very challenging to tune with multiple models working in unison (optimal tuning is tough) many parameters; some demos/results are very misleading (not necessarily good with generalization), operate under assumption the objects are perfectly detected; trained on specific videos.

Multiple-Object Tracking (MOT)

- Future directions: MOT research is already decades-old; most methods require offline trained object detector; need to adopt a generic "person" detector.

- Multiple cameras: major question how to fuse information from multiple cameras (can use traditional multi-modal data ensemble approach); for cameras with non-overlapping viewing regions, question about to re-identify. How to use known geometry of scene for inference (very much an open problem): one idea: project large inference problem to entire floorplan (in 2-d) and generate tracklets here; question, again, how to fuse data streams. (How to automate for new store geometries?)

- 3-d object tracking? Requires camera calibration, estimate poses and scene layout.

- Can save bandwidth in real-time by ignoring some feeds.

Multiple-Object Tracking (MOT)

- Future directions:

- MOT with scene/context understanding: first analyze image with scene understanding (could perform low-computation image segmentation), use results from scene understanding to provide contextual information and scene structure; this approach could be highly valuable (if not essential) for object tracking, gesture recognition, behavior classification, etc.

- MOT with deep learning: (still open) DL model can furnish a strong observation model that performs, say, image classification, object detection or SOT. Could parallelize strong SOT with DL for MOT (possibly).

- MOT with other CV tasks: symbiotic relationship between MOT and other conventional CV tasks (inference for one helps with inference for the other and vice versa): segmentation, human re-identification, face detection/recognition, action recognition, etc.

MOSSE tracker

- This algorithm is proposed for fast tracking using *correlation filter* methods. Correlation filter-based tracking comprises the following steps:

- (1) Assuming a template of a target object *T* and an input image *I*, we first compute the FFT of both the template *T* and the image *I*.

- (2) A convolution operation is performed between the template T and the image I.

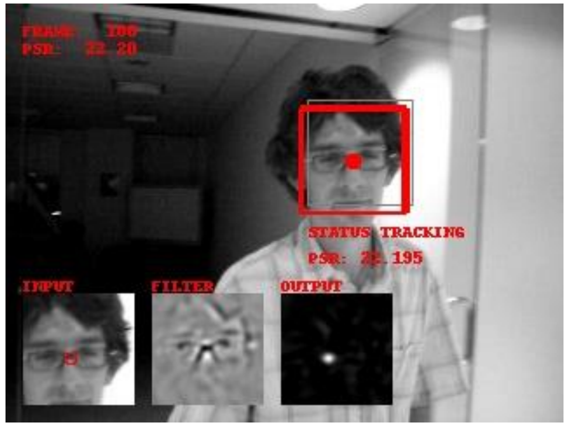
- (3) The results from (2) is inverted to the spatial domain using IFFT. The position of the template object in the image I is the max value of the IFFT response.

MOSSE tracker

The aforementioned correlation filter-based technique has limitations in choice of T. As a single template image match may not observe all the variations of an object, such as rotation.

Bolme et al., propose a more robust tracker-based correlation filter, called the Minimum Output Sum of Square Error (MOSSE) filter. In this method, the template T for matching is first learned by minimizing a sum of squared error as:

$$\min_{T^*} \sum_i \left| I_i \odot T^* - O_i \right|$$



SORT (Simple Online and Realtime Tracking)

SORT is a recent algorithm for MOT (Bewley et al., 2016)

In the problem setting of MOT, each frame has more than one object to track. A generic method to solve this problem has two steps:

(1) Detection: Detect object in frame

(2) Association: Once we have the detections, a matching is performed for similar detections with respect to the previous frame. The matched frames are followed through the sequence to get the tracking for an object.

SORT

Here are the (3) steps in SORT:

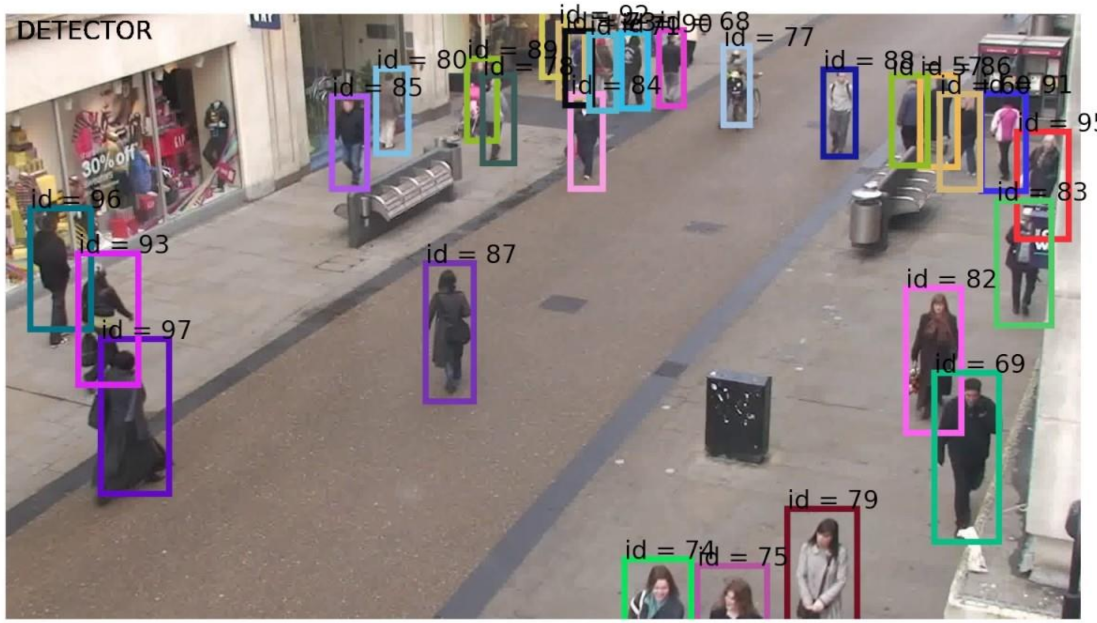
(1) In the paper they use Faster-RCNN to perform the initial detection per frame.

(2) The intermediate step before data association consists of an estimation model. This uses the state of each track as a vector of eight quantities: a box center (x,y), box scale (s), box aspect ratio (a), and their derivatives with time velocities. The Kalman filter is used to model these states as a dynamical system. If there is no detection of a tracking object for a threshold of consecutive frames, it is considered to be out of the frame or lost. For a newly detected box, a new track is started.

SORT

Here are the (3) steps in Deep SORT:

(3) In the final step, given the predicted states from Kalman filtering, as association is made for the new detection with old object tracks in the previous frame. This is computer using the Hungarian algorithm on bipartite graph matching.



Deep SORT (extension of SORT)

Simple Online and RealTime Tracking with A Deep Association Metric

Wojke, et al. (2017)

The authors integrate appearance information to improve the performance of SORT. This extension allows objects to be tracked through longer periods of occlusions (effectively reducing the number of identity switches).

Specifically, they integrate motion and appearance information using two metrics.

(*) For motion information they use the Mahalanabois distance between predicted Kalman states.

(*) To alleviate the problem of re-identification in the case of long-term occlusions they introduce a cosine metric applied to appearance descriptors of each identified object.

Multiple-Object Tracking (MOT) for Multiple Cameras

One possible workflow for MOT with multiple cameras using attribute-based person tracking.

*Source: IBM patent (2010) https://patents.google.com/ patent/US9134399B2/en

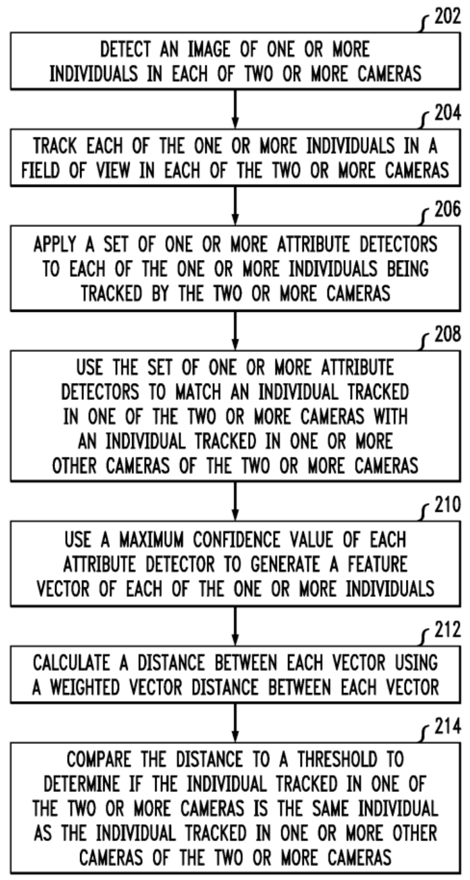
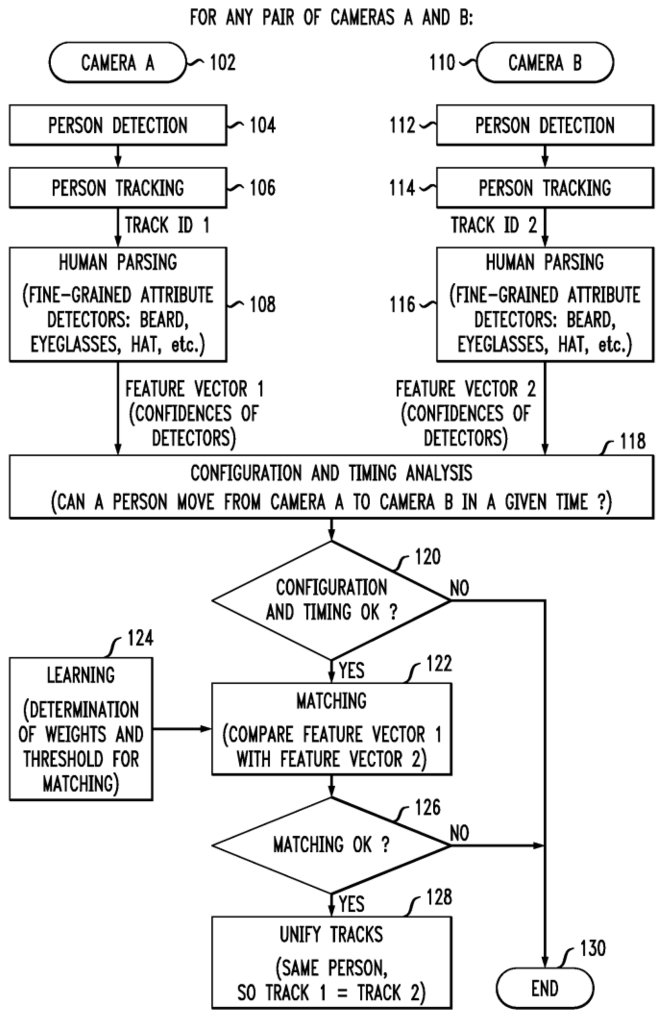

202
DETECT AN IMAGE OF ONE OR MORE INDIVIDUALS IN EACH OF TWO OR MORE CAMERAS
204
TRACK EACH OF THE ONE OR MORE INDIVIDUALS IN A FIELD OF VIEW IN EACH OF THE TWO OR MORE CAMERAS
206
APPLY A SET OF ONE OR MORE ATTRIBUTE DETECTORS TO EACH OF THE ONE OR MORE INDIVIDUALS BEING TRACKED BY THE TWO OR MORE CAMERAS
208
USE THE SET OF ONE OR MORE ATTRIBUTE DETECTORS TO MATCH AN INDIVIDUAL TRACKED IN ONE OF THE TWO OR MORE CAMERAS WITH AN INDIVIDUAL TRACKED IN ONE OR MORE OTHER CAMERAS OF THE TWO OR MORE CAMERAS
210
USE A MAXIMUM CONFIDENCE VALUE OF EACH ATTRIBUTE DETECTOR TO GENERATE A FEATURE VECTOR OF EACH OF THE ONE OR MORE INDIVIDUALS
212
CALCULATE A DISTANCE BETWEEN EACH VECTOR USING A WEIGHTED VECTOR DISTANCE BETWEEN EACH VECTOR
214
COMPARE THE DISTANCE TO A THRESHOLD TO DETERMINE IF THE INDIVIDUAL TRACKED IN ONE OF THE TWO OR MORE CAMERAS IS THE SAME INDIVIDUAL AS THE INDIVIDUAL TRACKED IN ONE OR MORE OTHER CAMERAS OF THE TWO OR MORE CAMERAS

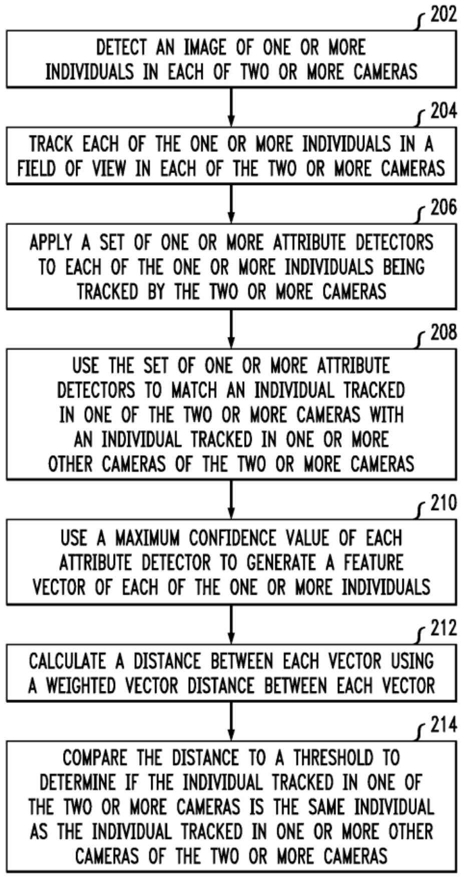
FIG. 1


FOR ANY PAIR OF CAMERAS A AND B:
CAMERA A
102
110
CAMERA B
PERSON DETECTION
104
112
PERSON DETECTION
PERSON TRACKING
106
114
PERSON TRACKING
TRACK ID 1
TRACK ID 2
HUMAN PARSING (FINE-GRAINED ATTRIBUTE DETECTORS: BEARD, EYEGLASSES, HAT, etc.)
108
116
HUMAN PARSING (FINE-GRAINED ATTRIBUTE DETECTORS: BEARD, EYEGLASSES, HAT, etc.)
FEATURE VECTOR 1 (CONFIDENCES OF DETECTORS)
FEATURE VECTOR 2 (CONFIDENCES OF DETECTORS)
118
CONFIGURATION AND TIMING ANALYSIS (CAN A PERSON MOVE FROM CAMERA A TO CAMERA B IN A GIVEN TIME ?)
120
CONFIGURATION AND TIMING OK ?
NO
124
LEARNING (DETERMINATION OF WEIGHTS AND THRESHOLD FOR MATCHING)
YES
122
MATCHING (COMPARE FEATURE VECTOR 1 WITH FEATURE VECTOR 2)
126
MATCHING OK ?
NO
YES
128
UNIFY TRACKS (SAME PERSON, SO TRACK 1 = TRACK 2)
130
END

Multiple-Object Tracking (MOT) for Multiple Cameras

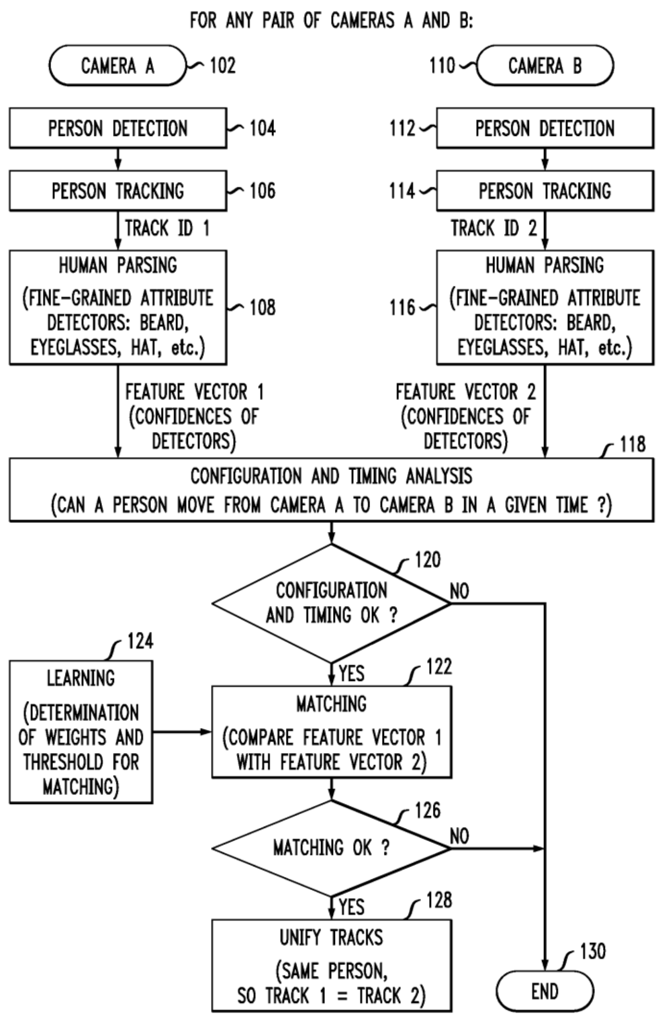
One possible workflow for MOT with multiple cameras using attribute-based person tracking.

*Source: IBM patent (2010) https://patents.google.com/ patent/US9134399B2/en



202
DETECT AN IMAGE OF ONE OR MORE INDIVIDUALS IN EACH OF TWO OR MORE CAMERAS
204
TRACK EACH OF THE ONE OR MORE INDIVIDUALS IN A FIELD OF VIEW IN EACH OF THE TWO OR MORE CAMERAS
206
APPLY A SET OF ONE OR MORE ATTRIBUTE DETECTORS TO EACH OF THE ONE OR MORE INDIVIDUALS BEING TRACKED BY THE TWO OR MORE CAMERAS
208
USE THE SET OF ONE OR MORE ATTRIBUTE DETECTORS TO MATCH AN INDIVIDUAL TRACKED IN ONE OF THE TWO OR MORE CAMERAS WITH AN INDIVIDUAL TRACKED IN ONE OR MORE OTHER CAMERAS OF THE TWO OR MORE CAMERAS
210
USE A MAXIMUM CONFIDENCE VALUE OF EACH ATTRIBUTE DETECTOR TO GENERATE A FEATURE VECTOR OF EACH OF THE ONE OR MORE INDIVIDUALS
212
CALCULATE A DISTANCE BETWEEN EACH VECTOR USING A WEIGHTED VECTOR DISTANCE BETWEEN EACH VECTOR
214
COMPARE THE DISTANCE TO A THRESHOLD TO DETERMINE IF THE INDIVIDUAL TRACKED IN ONE OF THE TWO OR MORE CAMERAS IS THE SAME INDIVIDUAL AS THE INDIVIDUAL TRACKED IN ONE OR MORE OTHER CAMERAS OF THE TWO OR MORE CAMERAS

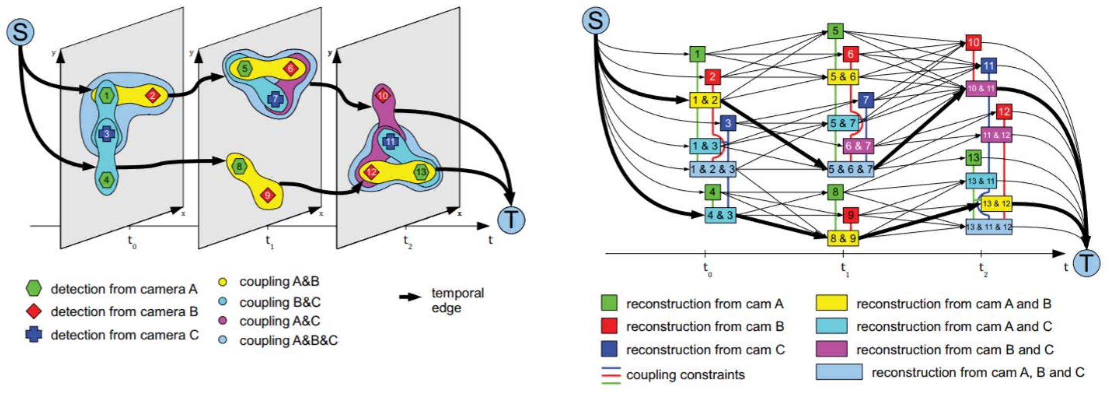
FIG. 1



FOR ANY PAIR OF CAMERAS A AND B:
CAMERA A
102
110
CAMERA B
PERSON DETECTION
104
112
PERSON DETECTION
PERSON TRACKING
106
114
PERSON TRACKING
TRACK ID 1
TRACK ID 2
HUMAN PARSING (FINE-GRAINED ATTRIBUTE DETECTORS: BEARD, EYEGLASSES, HAT, etc.)
108
116
HUMAN PARSING (FINE-GRAINED ATTRIBUTE DETECTORS: BEARD, EYEGLASSES, HAT, etc.)
FEATURE VECTOR 1 (CONFIDENCES OF DETECTORS)
FEATURE VECTOR 2 (CONFIDENCES OF DETECTORS)
118
CONFIGURATION AND TIMING ANALYSIS (CAN A PERSON MOVE FROM CAMERA A TO CAMERA B IN A GIVEN TIME ?)
120
CONFIGURATION AND TIMING OK ?
NO
YES
124
LEARNING (DETERMINATION OF WEIGHTS AND THRESHOLD FOR MATCHING)
122
MATCHING (COMPARE FEATURE VECTOR 1 WITH FEATURE VECTOR 2)
126
MATCHING OK ?
NO
YES
128
UNIFY TRACKS (SAME PERSON, SO TRACK 1 = TRACK 2)
130
END

Multiple-Object Tracking (MOT) for Multiple Cameras

- Several research groups have attempted to solve MOT with multiple cameras by rendering the problem as a Hypergraph (*cf.* Hypergraphs for Joint Multi-View Reconstruction and Multi-Object Tracking, Hofman et al.)

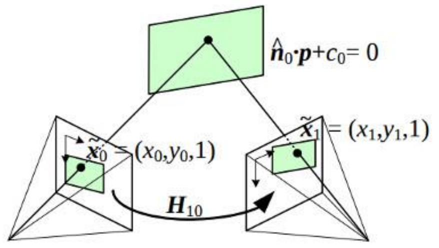

S
T
detection from camera A
detection from camera B
detection from camera C
coupling A&B
coupling B&C
coupling A&C
coupling A&B&C
temporal edge
reconstruction from cam A
reconstruction from cam B
reconstruction from cam C
coupling constraints
reconstruction from cam A and B
reconstruction from cam A and C
reconstruction from cam B and C
reconstruction from cam A, B and C

- Each detection in each camera frame becomes a node in the hypergraph; hyperedges denote potential couplings across different cameras; edge orientations in the graph connote temporal data.

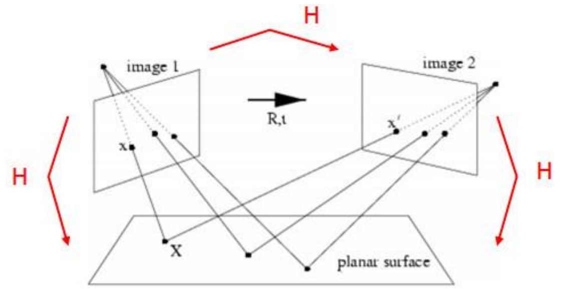
- The MOT problem is reduced to a constrained, min-cost flow graph; the tracking problem can be efficiently solved using a binary linear programming algorithm.

Multiple-Object Tracking (MOT) for Multiple Cameras

- Other attempted solutions to the MOT problem have relied on *homography*-based methods.



$\hat{n}_0 \cdot p + c_0 = 0$
$\tilde{x}_0 = (x_0, y_0, 1)$
$\tilde{x}_1 = (x_1, y_1, 1)$
H_{10}



Main idea: When a homography transformation is applied to images of an arbitrary 3D scene, the points that correspond to the plane will align, while the rest of the points will not.

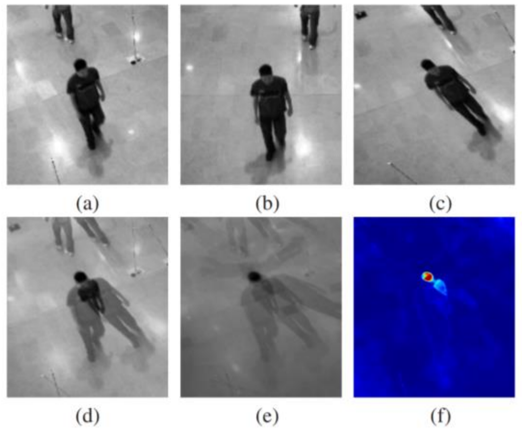


Figure 2. 2D patch detection demonstrated, for clarity, on a single, isolated person. (a,b) Two views of the same person. (c) Homography transformation is applied to image *b* to align points on the 3D plane at the head-top height with their counterparts in image *a*. (d) Image *c* overlaid on image *a*. (e) Overlay of additional transformed images. (f) Variance map of the hyper-pixels of image *e*, color coded such that red corresponds to a low variance.

Eshel et al., Homography Based Multiple Camera Detection and Tracking of People in a Dense Crowd

Multiple-Object Tracking (MOT) for Multiple Cameras

- Other attempted solutions to the MOT problem have relied on *homography*-based meth

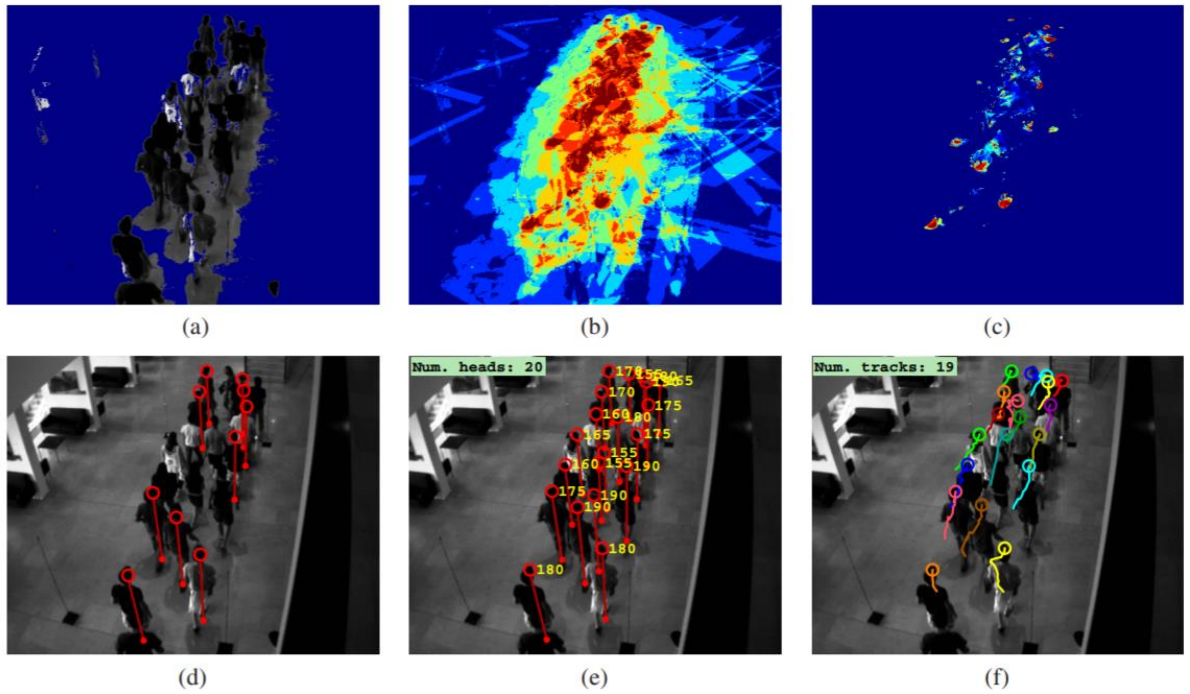


Figure 4. Intermediate results of head top detection. (a) Background subtraction on a single frame. (b) Aligned foreground of all views for a given height (color coded for the number of foregrounds in each hyper-pixel, where red is high). (c) Variance of the foreground hyper-pixels (red for low). (d) Detected head tops at a given height, and their projection to the floor. (e) The same as *d* for all heights. (f) Tracking results with 20 frame history.

Eshel et al., Homography Based Multiple Camera Detection and Tracking of People in a Dense Crowd

Multiple-Object Tracking (MOT) for Multiple Cameras

- Intel Open CV SDK supports: rcnn, yolo, AlexNet, VGG, Inception, MTT (baseline single camera) Etc.
- Acceleration with GPUs, traditional CV task support, MTT (single camera?), subject facial recognition, features (gender, age inference), optimization for Intel hardware.

Further Issues and Problem Considerations

- Integration of “stitching” for tracking across multiple cameras.
- Training robust person classifier (is a preexisting model sufficient?)
- Computation delegation (smart camera vs. cloud & on-line vs. offline)
- Identification of a group of consumers with single tag (e.g. family of shoppers).
- Method to utilize known geometry of store (for exclusion model).
- Fine-grained identification issues (e.g. is facial recognition, etc., a feature the model can use?)

Further Issues and Problem Considerations

- Meta-Data uses
- Gesture recognition
- Multi-modal resources for tracking (e.g. audio, sensors)
- Camera stream "parsimony" (e.g. when no individual detected in a frame, no need to process frame).
- Major challenge: Seamless integration of new inventory and store geometries for ambient computing; one-shot learning.
- Automation or semi-automation of system calibration in stores
- Primary component of "ambient computing" is MOT; with MOT alone one could potentially package meta-data pipeline using Movidius, etc., for commercial use (e.g. anonymous "heat map" tracking of consumer behavior).