# 1. Introduction

## Common Visual Object Recognition Tasks

* Captioning task: textual description
* Classification: is there a car in this image?
* Detection: where are the cars in this image?
* Activity: What is this person doing?
* Scene classification: is this an outdoor urban scene?
* Semantic segmentation: sky, building, person etc. One label per class
  + Instance segmentation: one label per object
* Object categorization: how to categorize any instance of an object type, e.g. cat or car

## Challenges

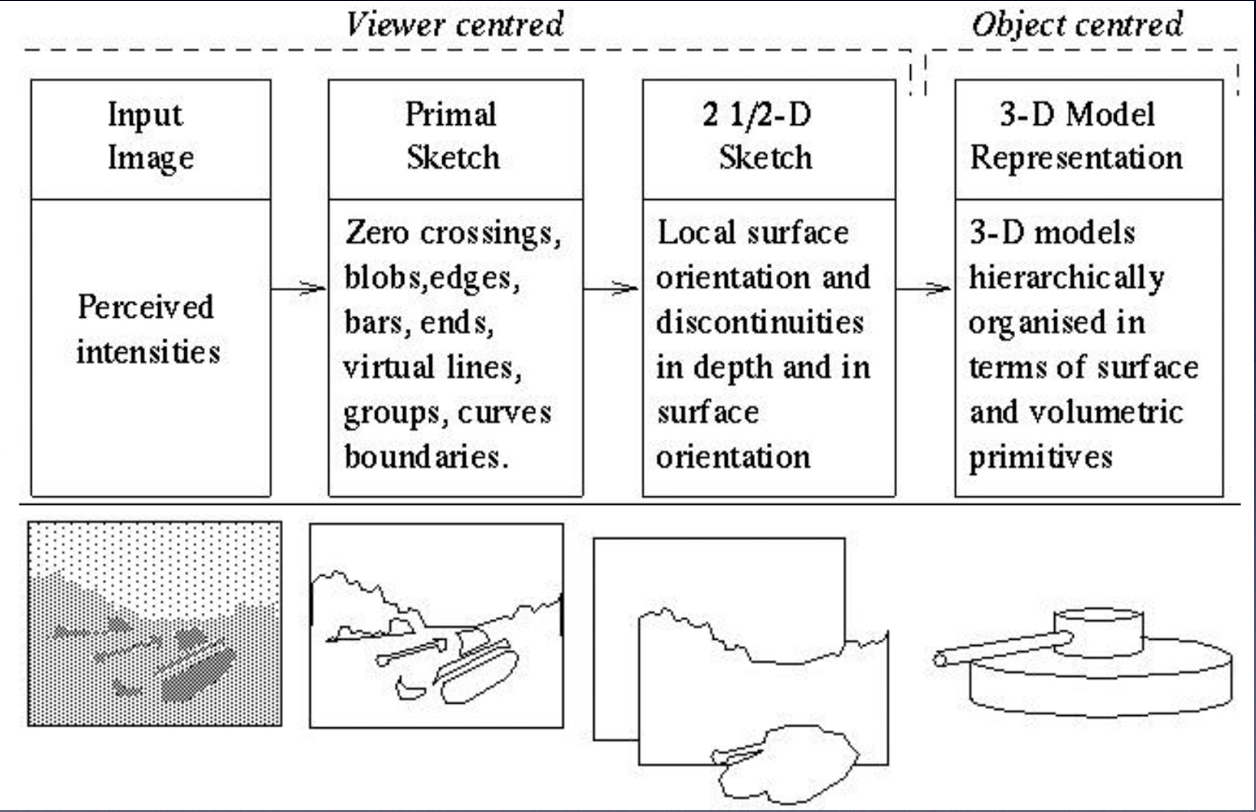
* Scale change: objects appear closer or more distant in the image
* Occlusion: objects are (partially) blocked in the image
* Viewpoint: objects are captured from many different angles
* Context: objects appear in diverse settings

## Applications

* Automotive safety: smart cars
* Face detection: different from face recognition (identity of detected face)
* Biometrics: fingerprints, faces
* Visual surveillance and tracking

## History

### Marr’s Vision (Stages of Visual Representation)



A proposed pipeline for how images are perceived and analyzed, i.e. vision:

1. Input image: perceived intensities
2. Primal sketch: zero crossings, blobs, edges, bars, ends, virtual lines, groups, curves, boundaries. Edge-based.
3. 2 ½D sketch: local surface orientations and discontinuities in depth and in surface orientation. Decorate with vectors and surface normal. 2D as it is still camera-centric, but also says something about the 3D arrangement of the objects/phases.
4. 3D model representation: 3D models hierarchically organized in terms of surface and volumetric primitives, e.g. spheres, blocks and cylinders.

This pipeline has been tried extensively and has proven to be brittle. Nowadays, different representations are used that are better suited for visual object recognition.

### 1960s – early 1990s

* Handcrafted models
* 3D geometry
* Hypothesize and align

#### Levels of Categorization

* Superordinate: animal
* Basic: dog, cat, cow
* Subordinate: German shepherd, Doberman

Definition of basic level:

* Similar shape: highest-level category for which members have similar shapes
* Similar motor interactions: people interact with its members using similar motor sequences
* Common attributes: significant numbers of attributes in common between pairs of members

While some categories clearly are defined by appearance attributes, a more common defining characteristic of linguistic categories is action attributes, e.g. a chair is something you sit on, a computer is an information processor etc.

### Mid 1990s – 2012

* Hand-crafted features
* Learned models
* Data-driven

#### Traditional Recognition Pipeline

Image pixels 🡪 hand-crafted feature extraction 🡪 trainable classifier 🡪 object class

Features are not learned and the trainable classifier is often generic, e.g. an SVM.

#### Image Features

Pieces of information about the content of an image; typically, whether a certain region of the image has certain properties. They are generally application dependent:

* Objects: local shape info, shading, shadows, texture, color
* Scenes: geometric layout, gradients, line segments, texture, color
* Actions: local shape info, motion, optical flow, texture, color

Global representations:

* Map image to a single vector based on overall characteristics, e.g. pixel intensities, greyscale/color histograms, bank of filter responses etc.
* + Works well for small articulations
* + Handles lower resolutions
* – Struggles with occlusions
* – Requires large training sets

Local representations:

* Describe component regions or patches separately, e.g. SIFT, chape context, superpixels, salient regions, geometric blur etc.
* + Handles occlusion
* + Handles rotations and view changes
* + Handles scale variations
* – Struggles with lower resolutions

#### Visual Classifiers

* Goal: from labelled training samples, learn parameters of a scoring/decision function that is likely to predict the correct label on test samples
* Typical assumptions:
  + Training and test samples drawn from the same distribution
  + Training labels are correct
* Nearest neighbor:
  + Take a vote from K closest neighbors in training set
  + Can learn features or distance measure
* Visual classifier:
  + Score is a linear combination of features
  + Examples: SVM, perceptron, naïve Bayes, logistic regression
  + Others learn features and the apply linear classifier, e.g. deep netowrks and random forests
* Structured prediction:
  + Score an interdependent set of labels, e.g. label body parts positions
  + Examples: structured SVM, CNN, graphical model algorithms

Most popular visual classifiers in computer vision:

* SVM
  + Linear for when fast training/classification is needed
  + Performs well with lots of weak features
* Logistic regression
  + Outputs a probability
  + Easy to train and apply
* Nearest neighbor
  + Hard to beast if there is tons of data, e.g. character recognition
* Boosted stumps or decision trees
  + Applies to flexible features
  + Incorporates feature selection
  + Powerful classifiers
* Random forests:
  + Decision tree variant that outputs probability
  + Good for simple features
  + Tons of features
  + Parallel training
* Deep networks / CNNs
  + Flexible output
  + Learns features
  + Adapt existing network (pre-trained) or train new with tons of data

### 2020 – Present

* End-to-end learning of features and models
* Requires labeled data availability
* Architecture design decisions, parameters

#### Deep Learning

Learning image features and classifiers what work jointly. Key ingredients are:

* Very large annotated datasets
* Large computational resources (GPUs, TPUs)

Classification error rate of ImageNet Large Scale Visual Regonction Challenge (ILSVRC) 2011 was 26%. With CNNs in 2012 it was 16% and is no less than 3%.

Common datasets:

* ImageNet: 22,000 categories, 14,000,000 images
* Microsoft COCO: 70 categories, 300,000 categories
* PASCAL. 20 categories, 12,000 images
* SUN: 5000 categories, 130,000 images

# 2. Feature Descriptors

## Terminology

* Feature: an aspect of an image that is useful in recognition
* Detection: where to sample
  + Can be replaced with sliding windows, i.e. exhaustive sampling
* Description: how to sample
* Observation: constructed by detection followed by description
* Descriptor: the result of an observation, a vector that can be compared to memory

Desirable properties of a descriptor vector:

* Invariance to nuisance parameters: illuminations, small positional shifts, region scale
* Discriminative power: such that different objects can be told apart (distance)
  + d(q, right\_model) = small
  + d(q, wrong\_model) = large

Descriptors estimate:

* Color: surface reflectance properties
* Texture: fine details, e.g. wrinkles and patterns
* Shape: coarse details, e.g. contours and depth boundaries
* In practice there is overlap between these, caused by the estimation process

Motivation for features, rather than training a deep network end-to-end

* Dataset size: training sets are generally too small to train a given network from scratch
* Computational power: it is time and resource intensive to train a deep network. Fewer parameters means shorter training time, e.g. hours instead of weeks

## Learned Descriptors

With access to a large set of labeled examples, it is possible to use machine learning to find good image descriptors. Efficient contrastive learning – finding similarities and dissimilarities – can be done using e.g. *triplet loss*:

* A baseline anchor is compared to a positive input and a negative input
* The distance from the anchor to the positive input is minimized
* The distance from the anchor to the negative input is maximized
* f(x) are descriptor vectors/network outputs
* Training uses mining for difficult triplets

Another option is to repurpose a network trained on some generic problem (e.g. object recognition) and use intermediate layer activations as pre-trained features.

* Weights for pre-trained deep feature extractors, a.k.a backbones, are available in most deep learning frameworks, including PyTorch
* By picking the output at one of the convolutional layers, a spatial feature map can be obtained
* By picking one of the fully connected layers, an image descriptor vector can be obtained
* Useful as generic descriptors in e.g. object recognition, subcategory recognition and scene recognition
* + A learned descriptor can improve performance significantly, compare to a hand-coded one
* – Domain shift risk if feature training was done on different type of data, which might reduce performance
* – Learned descriptor are computationally more expensive than designed descriptors

## Hand-Coded Descriptors

* Saves computations and is thus common for e.g. embedded systems application
* In practice, all designed descriptors have parameters that have been tuned, i.e. a form of learning is also used there

### Intensity Normalization

* A very simple descriptor, where the intensity normalized patch is described as:
* , where ,
* is a rectangular region cut out from an image and stored in a vector
* Insensitive to nuisance parameters such as illumination
* Similar normalization can be made when descriptors are compared instead, using zero mean normalized cross-correlation (ZNCC):
* However, it is generally desirable that descriptor comparison should be separable over descriptor dimensions for parallelization and speed purposes, so that e.g. search trees can be used, such as k-d trees together with approximative nearest neighbor (ANN) search
* Normalization is done spatially across the patch
* A common, related preprocessing for neural networks (NNs) is to standardize (or just center) the input with respect to the training set (or the batch)

### HOG Descriptor

* Image gradient: directional change in intensity (or color)
* The gradient of an image is a vector of its partials
* Histogram of oriented gradients
* A special case of the SIFT-descriptor, but adapted to dense grids

1. Gradient computation (with small filters):

)(x), ,

The image is convolved with the two filter kernels, one for each dimension, and the responses are stored in a gradient vector

1. Orientation binning:

The image is divided into small connected regions, cells, and for the pixels within each cell, a histogram of gradient directions is compiled. Each cell now contains K values, typically K = 9, one for each angular bin:

1. Descriptor blocks:

The cells are grouped into 2x2 blocks to account for changes in illumination and contrast:

The blocks typically overlap, so that each cell belongs to several blocks and thus contributes more than once to the final descriptor:

1. Block normalization to unit length: , where is some small offset/constant

The final descriptor is the concatenation of the block vectors

### Local Invariant Features

* Detector and descriptor pairs, i.e. to also decide where to compute the descriptor
* Difference of Gaussians (DoG):
  + Feature enhancement that involves the subtraction of one Gaussian blurred version of an original image from another, less blurred version of the original
* Scale invariant feature transform (SIFT):
  + Essentially an interest point detector (DoG) and a descriptor
  + The descriptor is a set 4x4 HOG cells that share a single common normalization
  + …plus some other tricks, such as spatial weighting and truncation
* Other local invariant features: SURF, BRISK, ORB, SFOP, FREAK

### Gabor Jet

* Weight maps of early layers in a CNN look similar on a diverse set of problems, given that the input is natural images
* They are often shifted, rotated and scaled variants of simple filters called wavelets
* Gabor jet are a family of filters that is often used to describe these types of filter
* A set of responses from filters that are oriented and localized wavelets
* The filters are complex:
* Essentially, it analyzes whether there is any specific frequency content in the image in specific directions in a localized region around the point/region of analysis
* Gabor jet is an example of a filter bank, i.e. a set of filters used together. Other common examples include derivative filters in multiple scales and wavelets
* Filter banks are particularly appropriate for texture representation and discrimination

### Binary Descriptors

* Local binary patterns are used in many descriptors to save memory and time
* The binary values represents the sign of an intensity difference
* Generally, local averages of differently sized regions are computed and the signs of the differences between region pairs are stored as a binary value
* Sign of intensity difference has monotonic illumination invariance
* A simple example is binary robust independent elementary features (BRIEF)
* Compares the signs of random pixel pairs in a patch and outputs a 256-bit representation, instead of e.g. 128 byte for SIFT
* Descriptors cannot be compared with Euclidian distance, instead:
* Very efficient to compute and matching is very efficient when supported by machine SIMD instructions
* Other examples: ORB, BRISK, FREAK, Ferns
* These are all useful together with binary NNs, e.g. XNOR-Net and random, forests

### Gist

* A global feature for images that is useful in scene categorization
* Motivation: perceptual studies indicate that scene category is recognized before semantic information such as objects and their relation
* Gabor jets in 4x4 grid (4 scales, 8 directions) on down-sampled images (128x128)
* Converts an image to 512 values

### Color Histograms

* Many different variants, a popular one is Blobworld:
  + Transform a region of interest from RGB to L\*a\*b\* color space
  + Use fixed, coarse binning of Lab\* space (5x10x10 bins)
  + The subset of 218 bins that can have non-zero values is selected, i.e. those that fall within the RGB gamut
* Color histograms completely discard spatial position in the region of interest, making them shift and scale insensitive
* A more recent feature is Color names:
  + Machine learning is used to classify regions as one out of eleven different colors
  + The result is a set of non-uniform decision regions in Lab space
  + The descriptors are computed by histogramming the individual pixels in the region of interest

### Shape Descriptors

* Used to capture depth discontinuities and/or object boundaries
* Typically requires that segmentation has been performed
* Classical edge-based descriptors are sometime also called shape descriptors, which is incorrect as they actually capture texture
* Motivations:
  + Handle background clutter in 3D scenes: patches cut out around features will have varying background, so learning can be used to determine when outer parts of a patch should be ignored
  + Handle large illumination changes: the gradient strength changes non-unfirmly and contrast may even be inverted

#### Contour SIFT

* Based on a detector that produces contours, e.g. MSER or MSCR
* Region shape is robust to changes outside the region. Thus, if the region shape can be described, this robustness will be inherited by the descriptor
* A descriptor is computed from the binary mask of the region, instead of the grey-scale patch
* The resulting patches are less descriptive, but are more robust to illumination changes and background clutter
* The “standard SIFT pipeline” is used, but with binary patches as input instead of greyscale input
* All parameters are retuned to maximize performance on the binary paches

Other common shape descriptors:

* Shape context descriptors
  + Takes a number of points on the contour of a shape
  + Computes histograms from a number of bins in log-polar space around regions of interest
* Fourier descriptors
  + Rather old, first published 1972, but still in use
  + Represent points (coordinates) along a closed shape contour as complex numbers
  + The time index is defined as the sequence index along the contour
  + Apply the Fourier transform on the resultant periodic signal
  + DC component (0 Hz-term, mean): corresponds to a shift
  + Phase: corresponds to a rotation
  + Scale changes results in uniform scaling of all values in the descriptor

# 3. Convolutional Neural Networks (CNNs)

## Machine Learning

* Deep learning revolution: essentially a special kind of machine learning
* Some objective is minimized
* Minimization is in terms of the expectation over some loss function L, i.e. the loss cannot be computed but only estimated
* Ideal: expectation over data distribution, but the true data distribution is not available
* Hope: sufficiently large empirical data (training set) gives the same parameters (empirical risk minimization)
* Hypothesis: test set drawn from the same distribution
* Models with high capacity/complexity memorize training set and do not generalize well over the test set, i.e. overfitting
* On the contrary, optimization is a direct minimization of an objective on data
* Machine learning problems:
  + Supervised learning: the system has access to both input and target output during learning, e.g. regression and classification
  + Unsupervised learning: the system has access to input but not target output, e.g. clustering and topic modelling
  + Reinforcement learning: the system learns by interacting with the environment, e.g. motor babbling and bootstrapping

### Linear Regression

* Many terms and concept similar to those in NNs
* Least square minimization of the mean squared error (essentially L2 loss):

= model parameters, = predicted value, = target value

* Instead of solving in closed form, consider using gradient decent:
  + Start with an arbitrary value for
  + Compute the gradient of the loss
  + Update by adding the (scaled) negative gradient: , where is called the learning rate
  + Repeat the first two steps until the loss is sufficiently low
* Substituting L2 loss into the generic update rule:
* In vectorized form:
* Variants of gradient decent:
  + Minibatch gradient descent: update based on a small batch of the training data, which usually gives faster convergence
  + Stochastic gradient descent: update based on a random training instance, which usually avoids local minima
* Not restricted to linear regression, as long as the model function and the loss is differentiable, i.e. the gradient can be calculated
* Underfitting/overfitting:
  + Model trained/optimized on training data
  + Evaluated on test data to estimate the generalization capability
  + Validation data can be used to optimize meta parameters

## Feedforward Networks

* Linear separability may be achieved for non-linearly separable data by introducing more features
* This may be done by applying a linear model not to but to representation of
  + Manually engineering using expert knowledge – feature engineering
  + Make the model sensitive to parameters such that learning them identifies a good representation – feature learning (the purpose of deep networks)
* A neural network (NN) is generally a directed acyclic graph (DAG), which can be seen as a function decomposition
  + In this case, it means that f is evaluated first and then the result is fed into g
* The length of this composition chain is the depth of the model (network)
* One layer: (linear discriminator):
* Two layers (additional feature representations): ],
* Input layer 🡪 hidden layers 🡪 output layer
* Instead only using linear scalar combinations between input and weights at a node, non-linearity can be introduced using an activation function f, which makes the node an artificial neuron:
* Network representation:
* Some loss function: , e.g. (L1 loss)
* The objective during training is to minimize the total loss:

### Activation Functions

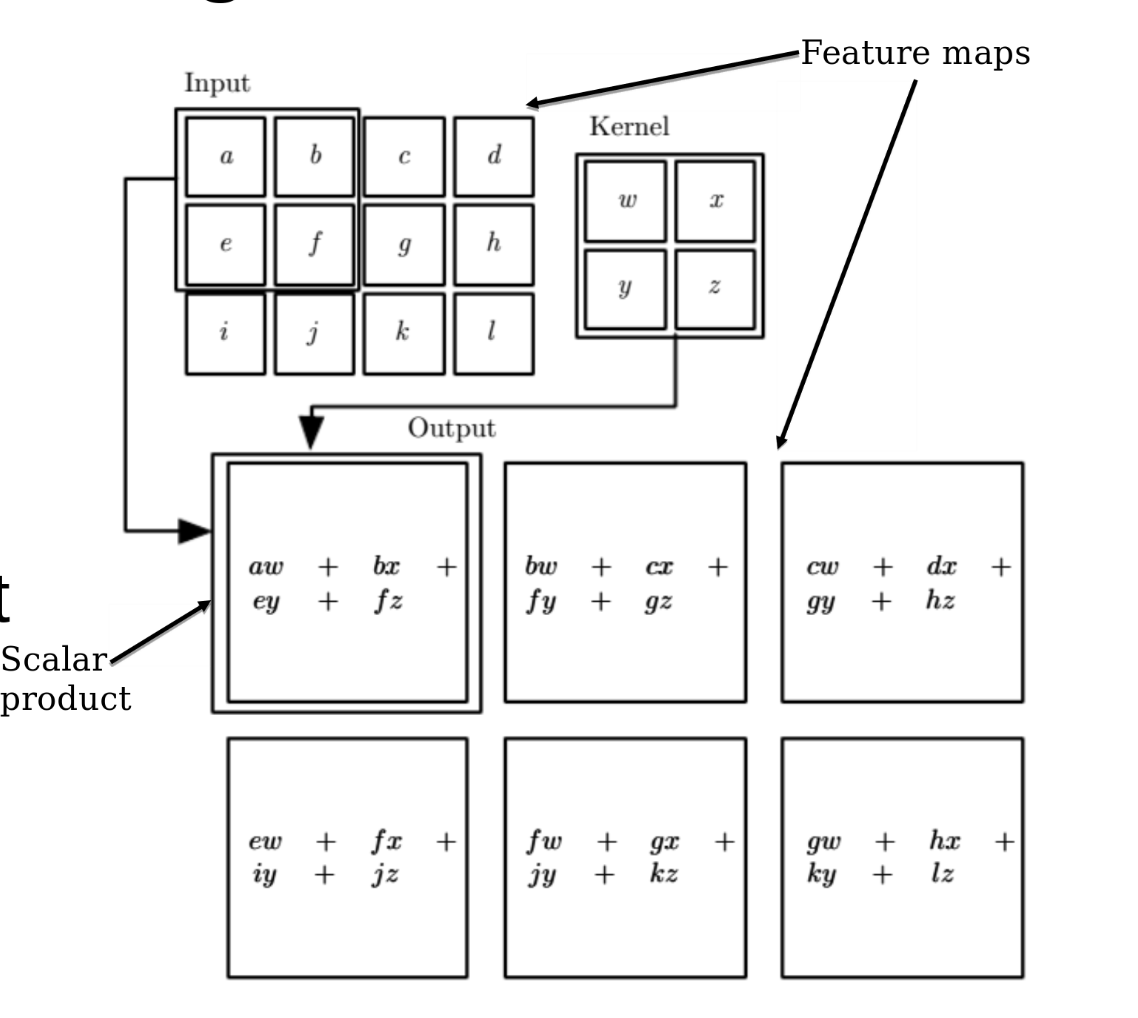
* The purpose is to introduce non-linearity
* Earlier, threshold functions were used, but these are now often approximated using some smooth function
* Logistic function: which gives
* Strictly speaking, the logistic function only solves binary problems. However, in many cases it is desirably to perform classification to multiple classes. This generalization is performed using the softmax function.
* Softmax layer: , ,
* The logistic function has some drawbacks, e.g. that it saturates in both ends, which may cause vanishing gradients. Instead, use ReLU:
* Rectified linear unit (ReLU): ,
* ReLU is the simplest form of function that is non-linear and not bounded in one unit. However, it is non-differentiable in the origin: use Softplus that is very similar but smooth in the origin
* Softplus: (logistic function)

### Backpropagation

* Feedforward networks – chain of differentiable functions – typically trained using gradient descent
* Computational problem: how to efficiently compute the gradients for all layers of the network, at the same time
* The standard algorithm for this problem is backpropagation, which is essentially a computational trick to compute all chain rules efficiently
* Gradient descent:
  + computed recursively using the chain rule
* Want to calculate the derivative of the total loss w.r.t. the first weights in the very beginning of the network, i.e.
* Propagate backward in the network, calculating the derivative of the total loss w.r.t. activations at intermediate layers using (f is activation function, z is input to f (linear activation), y is output from f (intermediate activation), x is input):
* Error in the output layer k:
* Error in a hidden layer j:
* Computing weight gradients:
* By storing and , significant computation efforts are saved

## Convolution

* An image is generally represented as a long vector, with one value for each pixel, which gives very large weight matrices (squared image size) in an NN
* Hence, applying networks directly to images requires large computational efforts and the resulting models are prone to severe overfitting as the capacity is far too large
* Convolutional neural networks (CNNs) are neural networks that use convolution in place of general matrix multiplication in at least one of their layers
* Suitable for data with known, grid-like topology
  + Time series
  + Image tensors
  + Medical data
* Convolution is an operation where a filter/kernel is applied to some input signal
* Continuous signal:
* Discrete signal:
* A convolutional matrix is band diagonal, sparse – contains mostly zeros
* Corresponds to a local operation, i.e. elements far way in the input will not be involved with the same scalar products
* Boundary conditions in 1D:
  + Same: output has the same size as the input, add zeros to matrix 🡪 Toeplitz matrix
  + Valid: only return scalar product values where the filter fits completely inside the signal, cut rows in matrix
  + Full: grow the signal, add extra rows in matrix
* In 2D: doubly block Toeplitz
* Convolution computation: essentially scalar products between the kernel and input matrix, where the kernel is shifted over (all) positions in the input



* CNNs strictly speaking employs correlation, not convolution, i.e. the filter is not flipped/mirrored/rotated. However, this does not matter, the parameters of the network are learned either way
* Images becomes feature maps in the network
* All operations are local and sparse
* One scalar product for each output
  + The support of that scalar product, i.e. the area of the input matrix where the corresponding kernel coefficients are non-zero, is called the receptive field
    - The region in the input space that a certain feature is looking at

## Convolutional Neural Networks (CNNs)

#### Motivations

* 1. Sparse (and local) interaction
  + Assumed that kernels are smaller than the input
  + Every output is not affected by every input, or vice versa
  + + Fewer parameters
  + + Lower memory requirements
  + + Better statistical efficiency
  + + Fewer operations
  + – Points that are far away in the input cannot influence the same outputs
  + Addressed by increasing the depth of the network, so that an output is indirectly connected to all input points
  + 🡪 Many layers required to cover the whole image
  + The more layers, the larger the effect of the receptive fields
* 2. Parameter sharing
  + Assumed that weights are tied, i.e. same weights are used between e.g. x1–s1, x2-s2 etc.
  + + Reduced storage requirements
  + However, the time complexity is the same
  + Sharing should sometimes be limited, e.g. in boundary regions and cropped images
* 3. Give equivariant representations, i.e. if something in the input is shifted it will also be shifted in the resulting feature map
* The concepts of locality and shared weights are independent, so the amount of weight sharing can easily be regulated: shared, unshared, tiled
* Bias terms
  + Locally connected and unshared – each output unit (node) has its own bias
  + Tiled convolution – share biases in tiling patterns
  + Shared convolution
    - Share bias
    - Often, separate bias at each location is preferred, to compensate for differences in the image statistics, e.g. systematically darker regions around boundaries

#### Network Layers

Two different terminologies in the literature:

* Complex layer terminology – each convolutional layer consists of three stages:
  + Convolution stage: compute linear combination z
  + Detector stage: compute non-linear activation y
  + Pooling stage
* Simple layer – each stage is considered as a separate layer
* *Pooling* is some summary statistics of nearby outputs
  + Max pooling: maximum output in rectangular region (mostly used)
  + Average pooling: average output of rectangular region (suboptimal)
  + L2-norm pooling: Euclidian norm of rectangular region
  + Weighted average pooling: weighted average of region based on e.g. distance from central position in image
* A function f is *invariant* under operation g if applying g to the input of f does not change its output
* A function f is *equivariant* under operation g if applying g to the input of f changes the output
* Equivariance is easily obtained for discrete shift variations, e.g. convolutions
* Trickier to obtain for rotation and scaling operations
* Max pooling is approximatively invariant to small translations
* Shift invariance: Input can be shifted slightly in position with the same max values propagated to the next layer. The border/outer nodes are however not really shift invariant as they are likely to change given a shift in the input.
* Rotation invariance: Input can be slightly rotated with the same max values propagated to the next layer
* Potential issues with max pooling:
  + Risk for underfitting, the model is too crude
  + Induces topological knowledge, there must be some understanding regarding the connectivity between certain kernels (shifts, rotations)
* Strided pooling: pool s pixels apart instead of every pixel (stride s)
  + + Improved statistical efficiency
  + + Reduced memory requirements
  + + Handling of varying input size, by adapting the stride to the ratio between the input size and desired output size
  + – Striding somewhat complicates the top-down processing, e.g. for back propagation
* Strided convolution: calculate convolution value for every s position, instead of every single position
  + Closely related to sequential convolution followed by down-sampling, e.g. in filter banks and wavelets. But in contrast, in those cases all convolution values are actually computed.
* Zero-padding:
  + Valid method successively reduces the feature map
  + All other methods add zeros to outer nodes to keep a fixed-size feature map
  + Strided convolutions successively shrinks the feature map, but they should still cover the entire original areas, necessitating zero-padding

# 4. Image Classification with CNNs

## Loss Functions

### Likelihood

The expected loss over some distribution is given as:

If the training data is drawn from the distribution p (density function):

Optimization should be so that the parameters maximize the probability:

, but second (prior) term is often unknown, so only the first (likelihood) term is maximized

### Maximum Likelihood Estimation (MLE)

Motivation:

* Assume Gaussian probability distribution:
* The ML estimator for is (assuming i.i.d.):
* This example is parametric due to the Gaussian distribution

In general:

* Assume a family of probability distributions that assigns a probability to any sequence X of N examples
* The ML estimator for is defined as:
* Assume that the examples are i.i.d., this can be rewritten as:
* Assuming Gaussian noise in P, the error is the sum of squares

### Conditional Log-Likelihood

* In supervised learning: learn a conditional probability distribution over target values y, given features x
* The assumption that the samples are i.i.d. yields:
* This corresponds to the expectation over the density of x, cross-entropy:
* Principled way to derive cost function (including L2 error):
* Sigmoid and cross-entropy balance each other: the derivative of a cross-entropy cost function using a logistic activation gives , i.e. a linear error between the target and the prediction (independent of the absolute value of y).

### Surrogate Loss Function

Assume e.g. 0-1 loss for class membership (OHE) surrogated/replaced with log-likelihood (cross-entropy). Motivations:

* Gradient decent does not allow for 0-1 loss (discontinuous) – cross-entropy is continuously differentiable
* Test error with 0-1 loss might be lower for training with cross-entropy than 0-1 loss – cross-entropy pushes classes apart even if 0-1 loss on training set is zero 🡪 larger margin and more robust decision surface

### Hinge Loss

* Maximum-margin classification (used in SVMs):
* Multi-class classification, one-vs.-one/all options:
  + Crammer & Singer:
  + Weston & Watkins:
* Hinge loss is non-differentiable 🡪 tricky to use in deep learning
  + Sub-gradients may be used, as for ReLU, i.e.

## Regularization

### Norm-Based Regularization

* Training of a NN can be regularized by adding some additional term to the error function
* L2-regularization – give preference to parameter vectors with smaller Euclidian norms (“lengths”):
* L1-regularization – give preference to parameter vectors with smaller absolute-value norms:

### Selected Regularization Techniques

* Dataset augmentation: generate new training data by systematically transforming existing data, e.g. by shifting, rotation or scaling of images
* Early stopping: stop the training when the validation set error begins to increase and backtrack to the previous set of parameters
* Bagging/ensemble methods: train several different (independent) models separately, then have all of the models vote on the output
* Dropout: randomly set a fraction of units to zero during zero

### Local Minima

* Deep networks always lead to local minima, related to the model identifiability problem
* However, it is not necessarily a problem as the cost is mostly similarly low 🡪 true global minimum generally not relevant
* To detect whether a local minimum causes problem, the norm of the gradient over time can be plotted. If the network produces bad results and the norm is almost zero, the local minimum might be a problem, but this is rarely the case.
* In high dimensions, local minima are very sparse, saddle points are much more common. This is due to the consideration of the Hessian matrix, where in a local minimum, all eigenvalues have to have the same sign which is very unlikely in high dimensions.

### Optimization

* For training, variants on gradient descent are often used:
  + Momentum-based methods
  + RMSProp (adaptive weights)
  + Adam
* These methods are usually base on stochastic subsets of training data:
  + Stochastic gradient descent (SGD) – single samples
  + Mini-batches – several samples
  + Drawn without replacement until next epoch
* Batch normalization is often used, e.g. zero mean and variance one

### Initialization

* Gradient-based learning might lead to wrong search directions or long trajectories as it is an iterative method
* Initialization in an area connected to a solution results in faster learning
* It is suboptimal to initial weights symmetrically or identically, as this only produces redundancy
* Initial weights should instead be large to break symmetry, but they should not be too large in order to avoid numerical issues (overflow)
* Determining initial values is costly
* A good heuristic is to draw weights from a Gaussian or uniform distribution
* Normalized (Xavier) initialization for m inputs and n outputs (here tanh is activation; add factor 4 for sigmoid activation):
* Other methods: random orthogonal matrices, sparse initialization etc. but these are more computationally expensive

### Batch Normalization

* Reduce the range of numerical values
* In deep networks, the simultaneous update of layers will have second, third, …order effects, which complicates the gradient update and may cause the learning rate to change wildly
* To reduce these effects, the linear activations z are replaced during learning for every batch (simplified whitening):
* This normalized activation is transformed back with moving-average parameters to retain information about the long-time trend:

### Minibatch Methods

* Batch (deterministic optimization): whole dataset
* SGD: single training samples
* Minibatch: subsets from training set
  + Should be large enough to exploit multicore architecture
  + Should be small enough to fit into memory
  + Size is often a power of 2: 32, …, 256
  + Size coincides with accuracy of e.g. gradient estimates
  + Number of minibatches coincides with regularization (robustness)
  + Repeated drawing: epochs

### Learning Rate

* Stochastic methods (SGD, MB) require a decaying learning, which enforces convergence and avoids oscillations that may cause that local minima are missed
* Formal conditions for convergence:
  + Sum of learning rates goes to infinity
  + Sum of squared learning rates is bounded
* For convenience a linear combination of initial and final learning rate is often used:

where k is the number of iterations and is the total learning time

* Rule of thumb:
  + Initial rate larger than what initial results suggests
  + Final rate at about 1% of initial rate
* How the learning rate is updated over time is called curriculum or schedule

### Momentum-Based Methods

* Gradient descent can be problematic due to curvature, small scales or noise in the error surface
* Introduces velocity, where gradient descent is considered as a particle with mass moving downwards, giving it momentum
* Velocity is an exponentially decaying moving average of gradients:
* If hyperparameter 🡪ordinary gradient descent
* gives about factor 10 speed-up

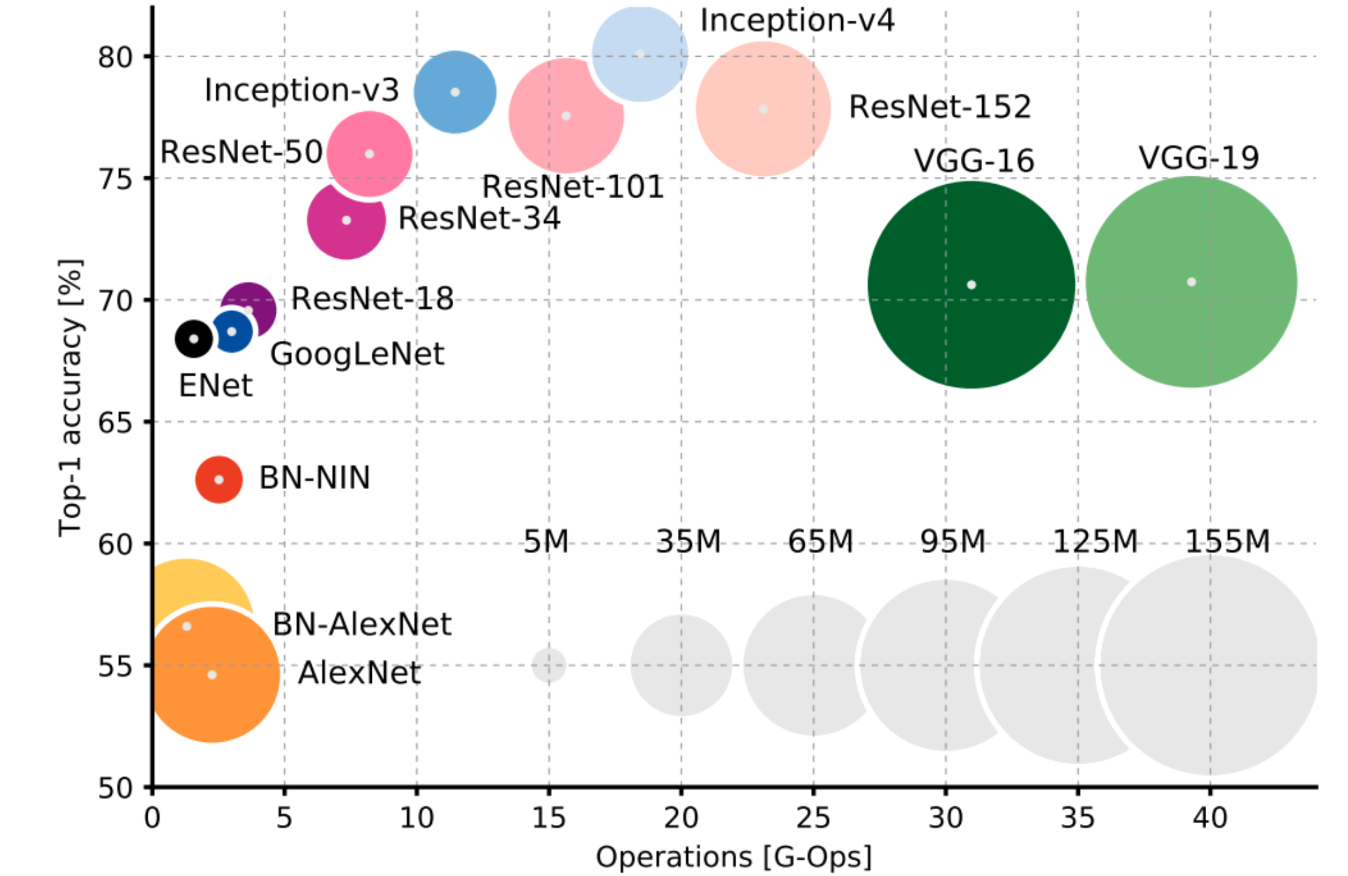
### Adam

* Based on RMSProp that uses adaptive gradient weight

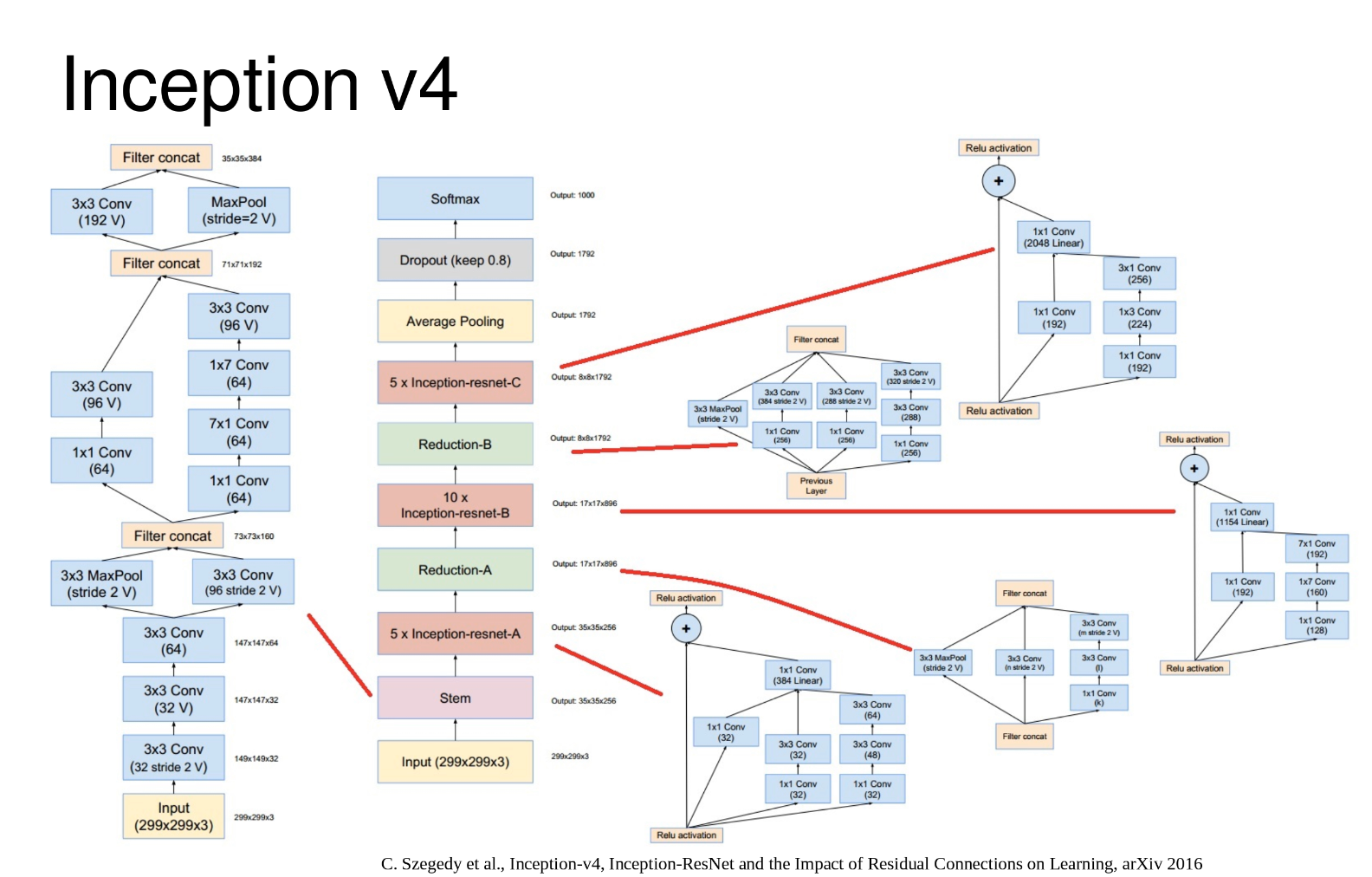
En bild som visar text

Automatiskt genererad beskrivning

## Network Architectures



* Simple CNN: a number of convolutional layers followed by a smaller number of fully connected layers and an output layer
* LeNet 5 (1998):
  + Quite simple and shallow architecture. Performs fairly well, but has two major shortcomings: less efficient pooling and vanishing gradients
  + Average pooling
  + Sigmoid/tanh activation
  + 60,000 training samples, 10 classes (MNIST)
* AlexNet (2012):
  + Quite similar architecture complexity to LeNet but has two streams so that the network can be split onto two GPUs
  + Max pooling
  + ReLU activation
  + Dropout regularization
  + 1,200,000 training samples, 1000 classes (ILSVRC)
* VGG (2018):
  + Very similar structure to AlexNet but is bigger
* ResNet (2016):
  + Far superior performance than AlexNet, but has about the same amount of parameters
  + Most commonly used “backbone” structure for downstream tasks
  + Idea: have shortcuts in the network
  + This effectively leads to a kind of residual learning where propagated and shortcutted intermediate results are compared
  + The ResNet-blocks stabilizes the learning, as there are direct connections between input and output
* Inception (2016)
  + Is a fairly complicated structure that is defined according to multiple modules, where each model is somewhat based on ResNet



* DAG-CNN
  + Multi-scale-CNN has shortcuts from all hidden layers to an output classifier
  + Each hidden layer has a direct shortcut to the output layer

## Adversarial Learning

Idea: make two different networks learn against each other to boost performance. Such a model of two networks are called Generative Adversarial Nets (GANs):

* Generator: fed with noise and generates a “fake” image
* Classifier: fed with real and generated images and determines binarily if an image instance is real or created by the generator
* The two networks are trained simultaneously so that they get as competent as possible at their respective task. The end-result is that the generator is able to produce images that are indistinguishable from the real ones and the classifier classifies these correctly at random, i.e. 50% chance.
* Loss function:
* Two expectation values: one over the data distribution (D) and one over the generator data (G)
* Min-max optimization: want to maximize loss over D and minimize loss over G
* Failures can easily be caused using simple fakes, which is true for all deep learning methods
* Universal adversarial perturbations: let the network generate a pattern that destroys classification results irrespectively of class. Furthermore, can be used on other networks as well, not only on the one it was trained on.

## Practical Aspects

* Making convolution efficient:
  + Parallel computation resources using e.g. GPUs
  + Clever convolution algorithms
    - Fourier transform – a convolution becomes a point-wise multiplication in the Fourier domain
    - Separability – a sequence of 1D convolutions reduces complexity and parameters (used in e.g. Inception-v4)
  + Deployment more relevant than training, as training is (often) only performed once
* Learning low-level features is the most expensive part of training
* If there is no access to these feature – e.g. via some backbone – there are some strategies to avoid supervised learning:
  + Random features – use to make decision about architecture
  + Hand-designed features – requires expert knowledge and are usually not as powerful as learned features
  + Unsupervised training of features – create subproblem where features are determined separately from some classification layer
* Approximative strategy for learning: greedy layer-wise pretraining, where lower layers are successively frozen during training
* Link to neural science:
  + Spatial map with 2D structure ~2D feature maps
  + Simple cells ~linear function of spatial region
  + Complex cells – pooling units with invariance
* Current CNNs span pathway retina-LGN-V1-V2-V4-IT
* However, a brain uses top-down feedback
* Foveal resolution (high in the center, lower in periphery) and saccades without counterpart in deep networks
* Given a noise stimulus, the patterns in response neurons in the hidden layers in a deep network are similar to those produced from Gabor functions (complex wavelets, varying coordinate system, scales, frequency, phase). This is not by design.
* Low level features are similar to edge detectors. If a CNN does not learn some edge detector, this is usually a bad sign.

# 5. Compound Descriptors, Metrics and Evaluation

* Feature engineering focuses on constructing observations, via detectors and descriptors
* Feature learning focuses on forming a memory and also matching, as network inference is a form of matching
  + A classifier CNN has the recognition memory in its weights
  + Activations are formed from convolutions of the input with these fixed learned weights, i.e. filters
  + This is equivalent to a sliding window scalar product 🡪 a type of template matching
* To perform other types of matching, descriptors can be arranged into more complex observations
* A way to better exploit the spatial arrangements of features is to pool these into compound descriptors or feature constellations
* Both local appearance and constellations contribute in human’s recognition process
  + Local appearance – how a local feature is perceived
  + Constellation – collection of local features which is perceived as a kind of higher-level, global feature

## Feature Constellations

### Local Feature View Clustering for 3D Object Recognition

* A view-based approach to object recognition
* A 3D object is represented as a collection of 2D views. This is in contrast to Marr’s vision, where 3D objects are represented as a collection of 3D primitives.
* In each 2D view, the coordinates of detected features are stored together with an affine transform – a geometric transformation preserving lines and parallelism – constraint, i.e. how much local features are allowed to move between views
* Not all views of an object are stored: during learning, similar views are clustered into a single view if they can agree on s feature arrangement under an affine transformation
* As 3D geometry is not explicitly used, views can represent both pose changes and articulation of an object, e.g. if a door is opened or closed, how a human moves its limbs
* In recognition, matching is first made by having each feature in a query image vote on possible matching views
* The most promising views are then verified by checking that they are consistent with the remembered affine transformations
* This system scales to several objects using ANN-trees, but if there are too many objects, the trees eventually become too large to be useful, necessitating other approaches

### Deformable Part Model

* A coarse global model called root filter
  + Used for template matching
  + Consists of a grid of HOG descriptors
* Connected to each root filter is a fixed number of part models with flexible spatial arrangement

### Bags of Features (Visual Words)

* Based on bag of words, using features as visual words
* A very coarse approach to allow flexible spatial arrangement – spatial locations of features are completely disregarded
* Can handle large magnitude of image and dataset sizes and are used to quickly index large datasets
* Essentially a histogram of occurrences of features in an image
* A visual word can be seen as quantized feature descriptor, e.g. a SIFT descriptor
* If a match is found, the spatial arrangement for the feature(s) should be verified in a subsequent step
* The resulting BoF vector is often into a ML algorithm or used in an ANN database lookup
* Biological counterpart to BoF: Anne Tresiman’s feature integration theory of human perception:
  + Features are processed in parallel and presence/absence of features matter, but not their location
  + In the subsequent attentional stage, spatial arrangement is utilized by directing gaze or attention toward a specific region of interest

#### Vector Quantization

* Vector quantization (VQ) allows the modeling of probability density functions by the distribution of prototype vectors
* In the original formulation of BoF, the descriptor space is VQ, using k-means clustering on a large training set, which gives k vectors (visual words)
* The clustering is done in whitened space: , C is covariance matrix
* Whitening is a generalization of standardization to nD and is a form of unsupervised metric learning. In this case, it is used to decorrelate the inputs as the estimated covariance of is the identity matrix.
* After whitening, each descriptor vector is quantized by approximated with the closest visual word
* The result of VQ is that the probability of visual words is somewhat equalized and the entropy is increased
* An alternative to k-means is to randomly draw case samples and use these as prototypes, which is much faster but not as good at increasing the general entropy

#### TF-IDF

* To weight the bins, each document (i.e. image) is represented as a vector of word frequencies, a BoW (word (feature) k, document (image) d):
* Term frequency inverse document frequency (TF-IDF) is used
* IDF weights further increases entropy but serves no purpose if the VQ is perfect. They also scale each dimension separately and can thus be seen as a specific choice of matching metric. If the result is to be used in ML, it is better to learn the weightings of the feature dimensions.
* Efficient image matching is done by a normalized scalar product, i.e. cosine similarity:
* To speed up matching, an inverted file can be used, which enables real-time matching on very large datasets. The inverted file is essentially a list for each feature that indicates in what frames it is present.
* If the visual word computations are simplified so that and IDF is omitted, the result is a histogram of visual word occurrences, i.e. a BoF
* The choice of k (number of words) is application dependent but typically, it is large and most values are zero. A smaller k gives a cruder appearance representation and a larger k induces the risk for overfitting, but for larger datasets, larger ks may be used.

#### Skipping of Interest Points

* Early BoF systems used detectors to decide where to compute descriptors
* Discovered: lower detection threshold 🡪 more descriptors in histogram computation 🡪 more informative BoF vector 🡪 better recognition results
* This is especially true for low-resolution images, where the number of detected points easily can be too low with standard detection thresholds
* However, a low detection threshold tends to give highly biased and noisy detection
* Solution: improve detection performance by random sampling of interest points instead of using a detector
* Nowadays it is popular to perform dense or gridded sampling to compute descriptors as input to BoF

#### Spatial Pyramids

* Can be used as an extension to BoF to reintroduce spatial information
* Images are divided into a fixed number of cells at different levels in a pyramid, e.g. 1, 4, 16…
* A BoF vector is computed separately for each cell and these are stacked together in the grids of the pyramid
* Larger grid cells are down weighted to compensate for the higher likelihood of matches there, usually by dividing matching scores with the cell width
* Even with a spatial pyramid, constellation information is not fully exploited in BoF approaches, so spatial verification may still be useful in subsequent steps
* The idea of spatial pyramids can also be used in CNNs by using spatial pyramid pooling (SPP) layers
  + CNNs as AlexNet assumes a fixed-size input image, this was earlier achieved by cropping or warping images of other sizes
  + However, this image resampling is avoided by the use of SPP and differently sized inputs of the same inputs can be used
  + Grid cells are defined as blocks that divide the image, not as pooling sizes or strides in terms of steps on a feature map
* BoF and spatial pyramids are global descriptors and have the same size regardless of the image resolution

### Descriptor Distances

#### Gaussian Least-Squares Distance

* Given a descriptor/query vector q in a query image, it has to be decided which prototype in memory () is most likely to correspond to the same world object using some definition of distance
* Assuming additive i.i.d. Gaussian noise on all elements, this can be posed as an MLE:
* So, the match with the smallest least square distance is most likely correct, assuming i.i.d. Gaussian noise
* If the vectors are normalized (NCC) there is a simple scalar product to calculate the distance:
* However, this approach is not always the optimal choice as the assumption of i.i.d. rarely holds in practice
* In a sense, this assumption of i.i.d. is made in all CNNs as activations are formed from convolutions with fixed learned filters. This is equivalent to a sliding window scalar product.

#### Chi² Distance

* Many descriptors, e.g. SIFT and BoF, are histogram-like in their nature
* Thus, their values typically follow the (discrete) Poisson distribution that has equal mean and variance:
* For large values of (e.g. 1000) a (continuous) Gaussian can approximate the distribution:
* Assuming independence, this leads to a negative log-likelihood:
* Estimating the variance by:
* The most likely match is then the one with the smallest Chi-squared distance:

#### Square-Root Distance

* Similar histogram measure to distance, a.k.a. the Hellinger kernel:
* It is a close approximation to and is faster if the square root is pre-computed, e.g. when using RootSIFT

#### Histogram Intersection

* Histogram intersection similarity is defined as:
* Used to be a popular similarity measure for histogram-type data as it is quite fast, e.g. in the spatial pyramid paper
* So far, all measures assume independence between bins, which is important as it allows for efficiency, e.g. in using ANN methods and search trees. However, it is essentially only an estimation.

#### Earth Mover’s Distance (EMD)

* In histograms, neighboring bins are typically correlated, e.g. in HOG neighboring bins corresponds to similar angles
* EMD is a way to match histograms with correlated bins
* Distance is considered as the cost of moving values in p to q
* The cost is considered as the distance times the amount to move
* To compute EMD, the LP transportation problem is first solved:
* can be interpreted as a joint PDF with marginals p and q
* The EMD is then computed as:
* The denominator is needed if histograms are computed from a variable number of samples, i.e. if p and q are normalized, the denominator will always be the same
* Generalization to the continuous case and other local norms than L1 is called the *Wasserstein distance*

#### Pyramid Match Kernel (PMK)

* EMD is quite expensive to compute and can instead be approximated using PMK
* Uses a scale pyramid over histograms where the bins are hierarchically grouped
* Coarser scales are down weighted to ensure Mercer kernel properties, which is mainly needed for SVM convergence. The kernel should be positive definite, which is attained by ensuring that the kernel is equal to an inner product in some space
* Similar to the concept of spatial pyramids for BoF, which was actually formulated using PMK

#### Ratio Score

* Given multiple query vectors/descriptors, it is favorable to rank their best matches to maximize the likelihood of a correct match
* Need to consider for both descriptor similarity and the risk of misclassification. Ratio scoring is an effective way to incorporate this risk.
* To account for the misclassification risk, matches can be scored according to the ratio between the best and the second-best match:

## Metric Learning

#### Mahalanobis Distance

* A descriptor should ultimately distinguish correct correspondences from incorrect ones
* A similarity metric can be learned by collecting known pairs of corresponding descriptors:
* d should be due to noise and nuisance factors
* To find a linear transformation that makes the noise equal in magnitude in all directions:
* Whitening transformation will make the covariance equal to the identity:
* Rotation also gives a valid solution:
* The rotation R should be selected so that the first few dimensions are where “things happen”, so that later, trailing dimensions hopefully can be discarded
* R can be found from PCA of the transformed descriptor space:
* Here descriptor covariance is used instead of noise covariance
* A final contraction operator selects the first k dimensions:
* For e.g. SIFT, is k x 128 truncated identity matrix
* Using this method, a linear mapping can be obtained that both improve the matching and reduce the descriptor dimensionality. For SIFT: 128 🡪 40 dimensions.

#### Linear Discriminant Embedding (LDE)

* Similar to Mahalanobis. For SIFT: 128 🡪 14/18 dimensions.
* LDE is computed by maximizing (w are linear projections):
* A is the covariance matrix for outliers and B the same for inliers
* Interpretation: want to find projections with large distance between outliers and small distances between inliers (correspondences)
* If B is inversible, J(w) is maximized by eigenvectors with large eigenvalues in
* B may be poorly conditioned, e.g. if the training set is too small. This may be handled by raising small eigenvalues in B using power regularization:
* can be interpreted as a threshold on the signal to noise ratio (SNR)
* Triplet loss is also a more general form of metric learning. While LDE can learn a single layer of weight, the triplet loss can be applied to an entire network to obtain deep descriptors.

## Performance Evaluation

### Classifier Evaluation

* True positives (TP): classified as positive and positive according to ground truth (GT)
* False positives (FP): classified as positive but negative according to GT
* True negatives (TN): classified as negative and negative according to GT
* False negatives (FN): classified as negative but positive according to GT
* The connection between these are often shown in a confusion table/matrix

#### Accuracy

* Perhaps the simplest measure, the frequency of correct outputs:
* This may also be used for multi-class problems, by treating each class a binary classification problem
* For multi-class problems top-n accuracy may be used to account for near misses
* Recognition problems are often unbalanced (fewer positives than negatives), and accuracy for such problems can be misleading, as changes in TP can drown in a large TN
* Two other measures that are invariant to skewed datasets, as normalization is done with GT set sizes, are the true positive rate (TPR) and the false positive rate (FPR):

#### ROC Curve

* Receiver operating characteristic
* ROC curves are invariant to skewed datasets, which is design choice and might not always be desirable
* The ROC curve is created by plotting TPR against FPR for various sensitivity/classification thresholds, i.e. the discriminant is moved

#### Precision-Recall Curve

* An alternative is to use precision and recall instead:
* The PR curve is created by plotting precision against recall for various sensitivity/classification thresholds, i.e. the discriminant is moved
* Precision looks only at reported positives, i.e. it is normalized by the total number of positives. Hence, PR are more informative than ROC if positives are rare (skewed dataset), e.g. in object recognition
* ROC and PR curves are used to evaluate binary classifiers across a change of the discriminant. Hence, classifier can be compared without committing to a specific threshold.
* The optimal discriminant direction is often application independent, but the actual threshold is not

#### Area Under the Curve (AUC)

* If a quality measure is to be used in optimization, a single measure is better than a curve
* A common way to summarize ROC and PR is to look at the AUC, i.e. the integral under the curve. This is also called average precision in case of PR curves.
* However, for practical reason, ROC and PR curves are often computed at discrete positions. To compute the integral of such a curve, they have to be interpolated.
* ROC curves are strictly monotonic 🡪 linear interpolation is reasonable
* Best practice for PR curves is to first calculate and interpolate the corresponding ROC curve and then converting the result to the PR curve
* Another option for ROC is the point of equal error rate (EER), i.e. where . However, AUC is generally better as it considers the whole curve.

#### Multi-Class Output

* Recognition algorithms often output class probability estimates for more than two classes
* To use ROC or PR curve, such problems must be converted to a set of binary problems. This can be done by treating each class as a binary classification problem and calculating the curves separately.
* A common way is also to use a class confusion table
  + The maximum output response from the classifier is selected and is used to index the row in the confusion table
  + Then the GT is used to index the column
  + The indexed celled in incremented by one and the procedure is repeated for each sample to fill in the table
* To retrieve TP, FP, FN and TN from a multi-class table, a single class output has to be selected. This is done by setting cell (x, x) to TP for that class, and set the adjacent row cells to FP and the adjacent column cells to FN.
* Summing these can give all measures previously discussed, per individual class

#### F-Scores

* Precision and recall can be combined into a single measure called F1-score by calculating their harmonic mean:
* The general, weighted F-score between precision and recall is defined as:
* A larger k gives fore important to the precision metric

### Detector Evaluation

#### Intersection over Union (IoU)

* GT assignment is needed for detectors with multiple outputs
* Basic idea:
  + 1. Assign 1-1 correspondence between GT and detection boxes. It is an example of an assignment problem, which is with the Hungarian method
  + 2. Compute the overlap error for all assignments and divide by the number of GT detections
* The overlap error is often computed from the IoU, a.k.a. the Jaccard index:
* The overlap error, a.k.a. the Jaccard distance, is then computed as:
* If A = GT and B = detection (for each pixel), this implies:

#### Repeatability Tests

* Repeatability can be used to evaluate feature detectors
* It uses some known geometric transformation between two views to check if the same region is detected in two different images
* The transformations can be given by using homography:
  + A point x should be transformed toa point x’ according to
  + In reality, regions are detected. If these regions are represented elliptically, they can be defined as a conic section matrix C
  + All points in the region should satisfy:
  + With m as the ellipse centroid and I as the covariance matrix, C can be explicitly defined as:
  + An elliptic region C(m, l) can be mapped through homography to a transformed region C’(m’, l’) using:
  + The transformation is applied both from the left and right side
* Procedure:
  + 1. Assign 1-1 correspondences from image 1 to image 2
  + 2. Compute overlap error
  + 3. Compute repeatability as: correspondences (with ) divided by the number of features (in the mutually visible region)
* Repeatability measures the probability that a feature will be detected again:
* It is not useful for non-rigid objects or categories. In those cases, annotated bounding boxes are needed as geometric constraints
* A complementary statistic is to simply count the number of corresponding regions, i.e. skip the division by number of detected features
* This is often better for object recognition: if each feature match casts a vote, the probability of a cluster forming by chance is low, so outliers can be tolerated
* If detection is followed by verification, the extra correspondence count can be removed later. The extra hypothesis generation only costs time.
* Descriptor matching generates tentative correspondences ordered according to matching score
* When GT is known, these can be evaluated with an inlier frequency (precision) curve (the higher curve the better). Good method for setting parameter of RANSAC.
* Bottom line: for detection and matching, both precision and total number of correspondences matter

# 6. Visual Object Detection

Difference between recognition and detection:

* Visual object recognition
  + Problem: is there a dog or a sofa in the image?
  + Input: image
  + Output: class probabilities
  + Evaluation metric: accuracy
* Visual object detection
  + Problem: where is the dog and the sofa in the image?
  + Input: image
  + Output: bounding boxes in the image with coordinates (x, y, w, h), sometimes together with confidences
  + Evaluation metric: IoU
* Often, detection is often followed by recognition, e.g. in face recognition
* Object detection focuses on object search: where is it?
* Typically, many candidate regions (patches) are selected and tested for object presence/absence
* Main idea: build templates that quickly differentiate object patch from background patch. The output from the comparison corresponds to binary classification problem: is the object located in the given patch?
* To simplify evaluation, it is common to agree on an IoU threshold for a true detection, e.g. 0.5. Precision and recall are then calculated at this specific threshold.
* It is also common to vary the IoU threshold to create a PR curve and calculate the average precision (AP), i.e. the AUC
* A good score requires both high precision and recall
* Challenges in object detection:
  + May be hard to model an object class:
    - Object instances of the same class may be very diverse
    - It is not always clear where an object should be delineated, i.e. where the bounding box (BB) should be located
  + May be hard to model non-object classes:
    - Bad BB localization
    - Confusion with dissimilar/similar objects
    - Miscellaneous backgrounds

## Object Detection Process

Object model specification 🡪 hypothesis generation 🡪hypothesis description and scoring 🡪 detection refinement

### Object Model Specification

* What parameters to use and how they should be used. Methods:
* Statistical template in BB (HOG-based patch template):
  + Object (x, y, w, h) in image
  + Features defined w.r.t. BB coordinates
* Articulated parts model:
  + More complex constellation model
  + Object is configuration of parts and each part can be detected
  + Hybrid model:
    - Use templates and constellations

### Hypothesis Generation

* Decide where to test the decided-upon object model, i.e. propose a set of test locations to the model. Methods:
* Sliding window (gridding):
  + Use fixed-size window to crop out patches at each location and scale
  + Each patch is separately classified
* Voting from patches:
  + Extract local features/interest points, e.g. patches or SIFT
  + Convert features to visual words, a.k.a. matched codebook entries, to index a memory
  + In the memory, auxiliary information is looked up, e.g. vector pointing to object center, label for the object type or object pose
  + The auxiliary information is converted to absolute values to be used in a continuous voting space
  + False matches tend to appear at random position, while true matches tend to form clusters
  + By performing clustering in the voting space, object hypothesis’ can be found
* Region proposal methods:
  + Use over-segmentation to generate many small regions with consistent local appearance called superpixels
  + Successively merge similar regions and search for blobby image regions that are likely to contain objects

The most common methods compared:

* Sliding window:
  + Comprehensive search over position, scale and sometimes aspect ratio
  + Typically, 100,000 candidates
  + Simple
  + Repeatable
  + Even with many candidates, a good fit to an object may not be found
  + Speedup through convolution often possible
  + Speedup through striding is possible, but detections may be missed
* Region proposal:
  + Region search guided by image contours/patterns with varying aspect/size
  + Typically, 2-10,000 candidates
  + Random, non-repeatable
  + Often requires resizing patches to fit a fixed size
  + More likely to provide candidates with a very good object fit
  + More recent CNNs can learn to output region proposals in the for of BBs

### Hypothesis Description and Scoring

* Compute the chosen features:
  + HOG-based template: extract HOG features and compare to a large template
  + CNN features: run the feature extraction network and input the results to classifier

### Detection Refinement

* Even though the task is now complete, it is very common to refine the detections to enhance performance
* Often, there are many detections for the same objects. The overlapping detections with lower score can be therefore be suppressed. Thereafter it is common to perform BB regression to predict a better location and shape using the remaining BB, used in e.g. DPM, R-CNN and SSD.
* Another method is to predict an object mask, using Mask R-CNN. It adds an additional network layer to reintroduce full-pixel resolution and outputs a binary mask for each object together with the class prediction.

## Object Detection Systems

### Viola-Jones Face Detector

* Exploits two key strategies:
  + Simple, super-efficient, but useful features
  + Early rejection (cascaded classifiers)
* Image features are a combination of binary (black and white) box filters (≈Haar wavelets):
* Regions of any size can be computed efficiently in constant time, using integral image 🡪 no need to scale images
  + Quick way to sum pixels in any rectangle
  + An integral image has pixels where each pixel value is the sum of all input pixels to the left and above it
  + Convert image I(x, y) to an integral image s(x, y) where
  + s(x, y) is computed using a small recursive filter:
* The detector uses mining of a large library of candidate filters considering all possible filter parameters: position, scale and type 🡪 180,00+ possible features associated with each 24 x 24 detection region
* AdaBoost algorithm can be used to both select informative features and to form the classifier
* Instead of evaluating the entire feature set at test time, small groups of features are selected to be run on all possible sub-windows. These groups are fed into a sequence of binary classifiers. The first classifier rejects many regions, but those accepted are passed to the next classifier in sequence 🡪 detection cascade
  + Each classifier should have a high FPR
  + Furthermore, the classifiers should be complementary, i.e. focus on different aspects of the problem. For this, boosting can be used, i.e. create a strong classifier of multiple weak ones.
* AdaBoost:
  + Select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of a weighted error.
  + Next, the examples are reweighted according to their errors so that previously misclassified examples are given more emphasis in the future
  + Then, the threshold for the remaining features are updated and a new best feature is selected iteratively
* Creating the detection cascade:
  + Make a chain of very fast classifiers
  + Bias early classifiers to have a high FPR
  + Positive decisions are passed on to the next classifier in the cascade
* Original training: 5000 positives and 350,000,000 negatives.
* Real-time detection using a 38-layer cascade with a total of 6000 features

### Dalal-Triggs Pedestrian Detector