Automated Feature Extraction with Machine Learning and Image Processing

Prof. Dr. Stefan Bosse

University of Siegen - Dept. Maschinenbau
University of Bremen - Dept. Mathematics and Computer Science

Advanced Object Detectors and Classifiers

How can we find, localize, and classify objects in engineering images?

Advanced Object Detectors and Classifiers

How can we find, localize, and classify objects in engineering images?

What are the challenges and pitfalls?

Advanced Object Detectors and Classifiers

How can we find, localize, and classify objects in engineering images?

What are the challenges and pitfalls?

Which models exist already - are they suitable?

Object Recognition in Images

Classifying the entire image containing one object ⇒ Simple Task (e.g., by using CNN or ANN models)

Object Recognition in Images

Classifying the entire image containing one object ⇒ Simple Task (e.g., by using CNN or ANN models)

Finding (detecting) and classifying of (multiple) objects ⇒ Challenging task!

Object Recognition in Images

- Object classifiction output: A discrete class label $c \in C$
- Object localization output: A position, i.e., a bounding box $b \in \mathbb{R}^n$, or a centre point $p_c(x,y,..)$
- Object detection output: Probability that there is an (or multiple) object(s)
- Object recognition: Class and bounding box
- Region proposal: Within a ROI there can be an object (foreground) or the ROI contains no object (background)

Object Recognition in Images

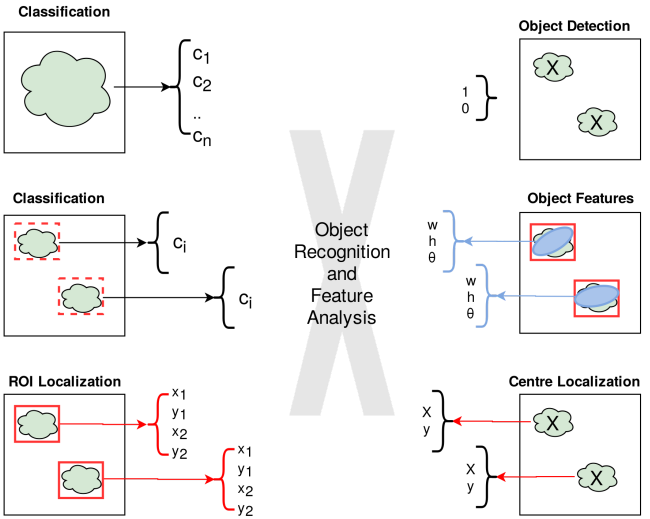


Fig. 1. Object classification, localization, detection, and higher-order feature prediction (e.g., geometrical features)

Object Recognition in Images

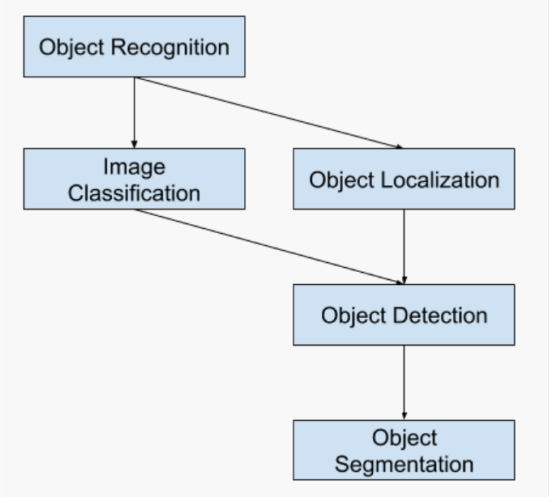
How to find objects in an image? The search parameter space is very big and high-dimensional!

- This is a chicke-egg problem: To find relevant regions we need firstly some understanding (i.e., classificaion) of objects, and than we can estimate the bounding box.
- Using generic features like edges, geonmetries (closed polgon paths), color boundaries, or colour similarities can help to find ROIs

Object recognition is a hybrid classification and regression problem!

- Different training loss (error) functions must be used to adress different feature output classes

Object Recognition in Images



[machinelearningmastery.com/object-recognition-with-deep-learning/]

Fig. 2. Overview of Object Recognition Computer Vision Tasks: Taxonomy of object recognition

Region Proposal

- First we need region proposals identifying Regions-of-Interest (ROI) ⇒ A (typical rectangular) bounding box region with a probability $\rho > \rho_{thres}$ that this region can contains
  1. any interesting object (just being foreground), or
  2. containing a specific class object, or inverse
  3. containing no object (background).

Many spatially distributed multi-object classifier models consist of two stages: region proposal and classification

Workflow

Given an input tensor of size CxHxW constructed from pixel values of some image ...

1. Identify content of interest;
2. Locate the interesting content;
3. Partition input (i.e. pixels) corresponding to identified content.

[A. Brown, 2017, NVIDIA]

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

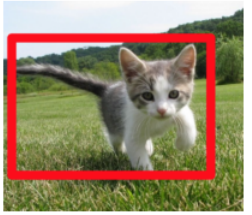
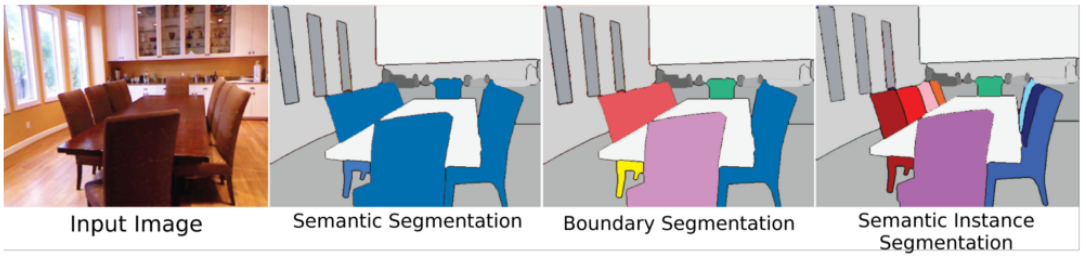


Image Segmentation

- Localization provides commonly dynamic (grid-less) bounding boxes
- Segmentation partitions images into static squared or rectangular regions intially unclassified, finally associated with an object class (and background) ⇒ the smallest segment is an image pixel!
- Semantic segmentation assigns pixels (or segments) to semantic classes, instance segmentation distinguishes different instances of one class:

[A. Brown, 2017, NVIDIA]



Region-proposal and Region-based Networks

- Region-propsale networks (RPN) deliver a map of bounding boxes and simultaneously predicts object bounding boxes and objectness scores at each position.
- Each bounding box is assigned a probability that the region conatins an object feature (or not)
- RPN is a fully convolutional network, which is trained in an end-to-end fashion, to produce high quality region proposals for object detection using Fast R-CNN

There is an imbalance in the false positive and false negative prediction rates: Higher FP is okay, higher FN is bad (missing objectes, e.g., smaller)!

Region-proposal and Region-based Networks

- Region proposoals (boxes) origante in **anchor points**

[https://towardsdatascience.com/region-proposal-network-a-detailed-view-1305c7875853]

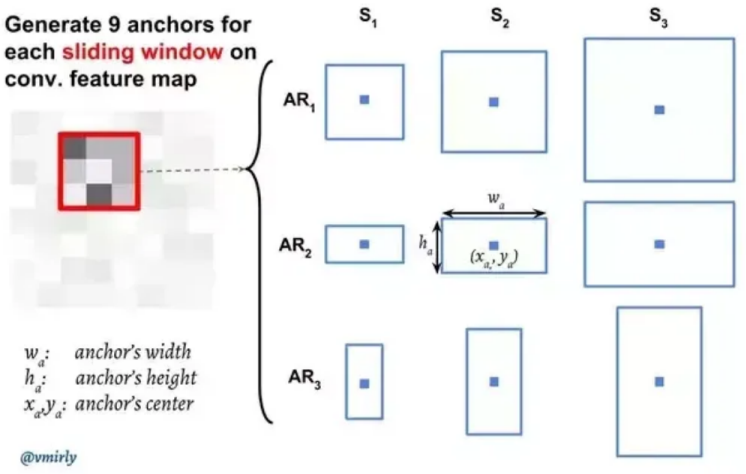


Fig. 3. Anchor boxes generation

Region-proposal and Region-based Networks

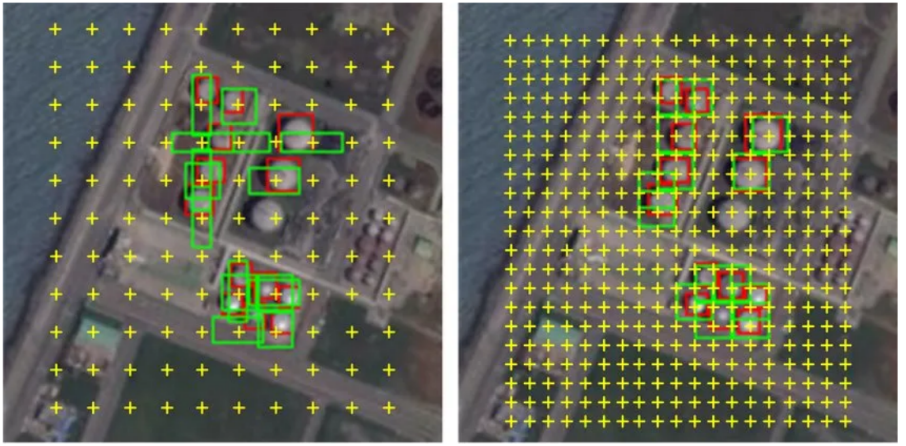


Fig. 4. Examples of region proposals originating from fixed spaced anchor points

R-CNN Model Family

The R-CNN family of methods refers to the R-CNN, which may stand for “Regions with CNN Features” or “Region-Based Convolutional Neural Network,” developed by Ross Girshick, et al.

This includes the techniques:

- **R-CNN**,
- **Fast R-CNN**, and
- **Faster-RCNN** designed and demonstrated for object localization and object recognition.

R-CNN

The R-CNN model is comprised of three modules; they are:

- Module 1: **Region Proposal**
  - Generate and extract category independent region proposals, e.g. candidate bounding boxes.
- Module 2: **Feature Extractor**
  - Extract feature from each candidate region, e.g. using a deep convolutional neural network.
- Module 3: **Classifier**
  - Classify features as one of the known class, e.g. linear SVM classifier model.

- The anchor boxes were created a priori in best hope, but these are dummy boxes that are different from the actual object of interest.
- Also, there might be many boxes which are not having any object in it. So we need to learn whether the given box is foreground or background, at the same time we need to learn the offsets for the foreground boxes to adjust for fitting the objects.
- These two tasks are achieved by two convolution layers on the feature map obtained from the backbone network. Those layers are rpn_cls_score and rpn_bbox_pred and the architecture looks like below.

- Once these fg/bg scores and offsets are learned using convolution layers, some portions of fg and bg boxes are considered according to confidence scores.
- The offsets are applied to those boxes to get the actual ROIs to be processed further.
- This post-processing of anchor boxes using offsets is called **proposal generation**.
    - These final proposals are propagated forward through the ROI pooling layer and fc layers.

Image Warping

- **Scaling of images to a normalized image (tensor/volume) size**
- In order to extract features from a region of a given image, the region is first converted to make it compatible with the network input. More precisely, irrespective of the candidate region's aspect ratio or size, all pixels are converted to the required size by warping them in a tight bounding box.
- A CNN convolution operation alaways applies a kernel of fixed size to an input matrix of fixed size

CNN Frameworks

ConvNet

https://cs.stanford.edu/people/karpathy/convnetjs/

- Pure JavaScript framework
  - Uses 3D volumes (with linear typed arrays)
  - No matrix/tensor algebra
- Supports:
  - Convolutional layer
  - Fully connected neural network layer
  - Softmax layer
  - Pooling layer
  - No deconvolution layer (expansion)

TensorFlow

- Native C/C++ implementation
- But tensorflow.js in JavaScript
- Split into front-end and back-end stages
- Different back-ends (CPU, GGPU, ..) supported by the same model architecture and configuration
  - But only selected/specific configuration are efficiently processed by a particular GGPU
- Uses always matrtix/tensor algebra
- Programming entry can be Python (but using native code libraries)
- Hard API break from version 1 to 2

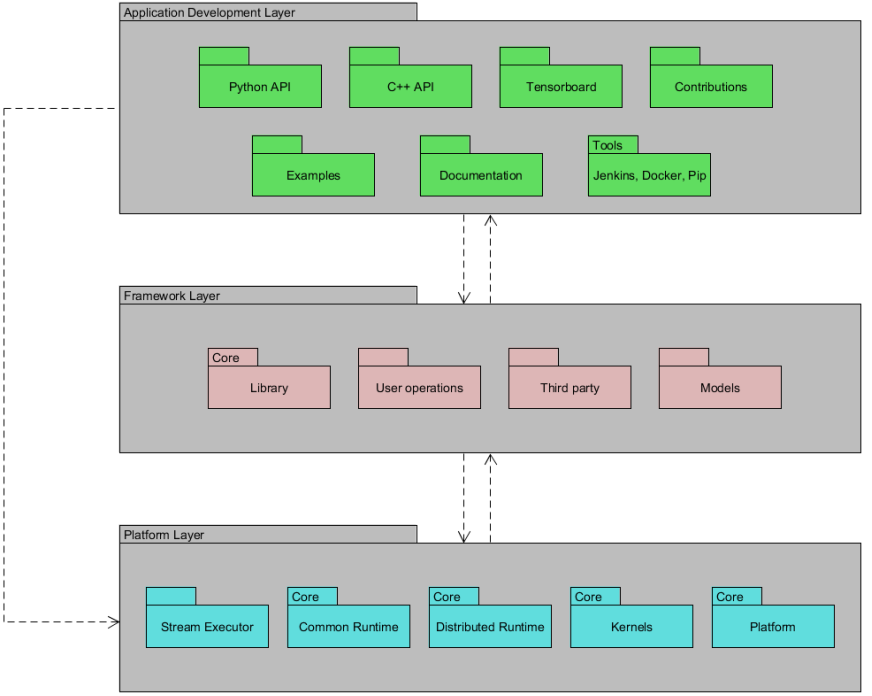


Fig. 5. Structure model of the TensorFlow

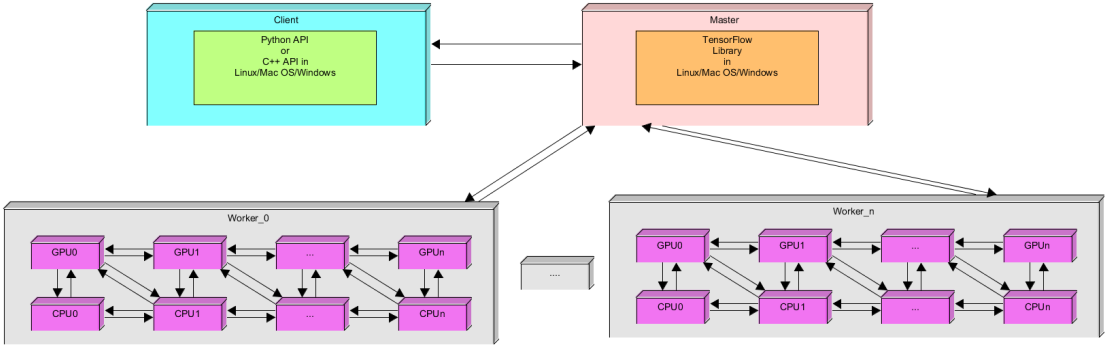


Fig. 6. Runtime Platform Model

A simple way to understand and implement Object Detection from scratch, by pure CNN.

See also:

https://tree.rocks/a-simple-way-to-understand-and-implement-object-detection-from-scratch-by-pure-cnn-36cc28143ca8

The idea: If you have ever used a Digital single-lens reflex camera (DSLR) before, you should notice the viewfinder is interesting.

- Let us consider a camera
- Points of interest are called “focus points.” like in camera's view finder
- The object exists probability: We think about CNN as the lens in a camera and focus points as the “object exists probability” points

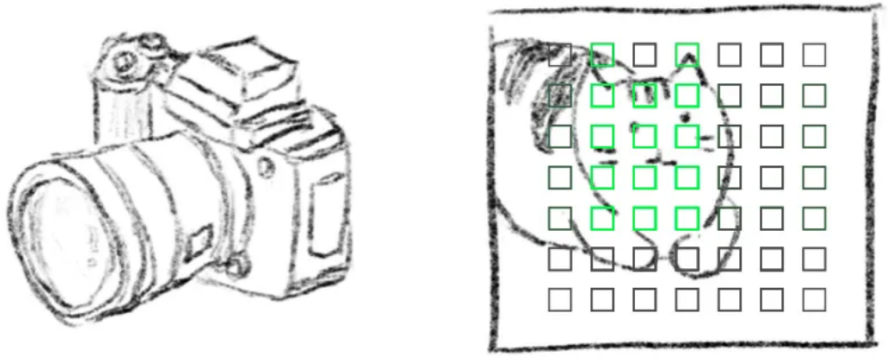


Fig. 7. An image viewn by a camera can be segmented in a matrix of focal points, some covering an object ⇒ ROI proposals

- An object classification is a network with an input image covered mostly by the object to be classified

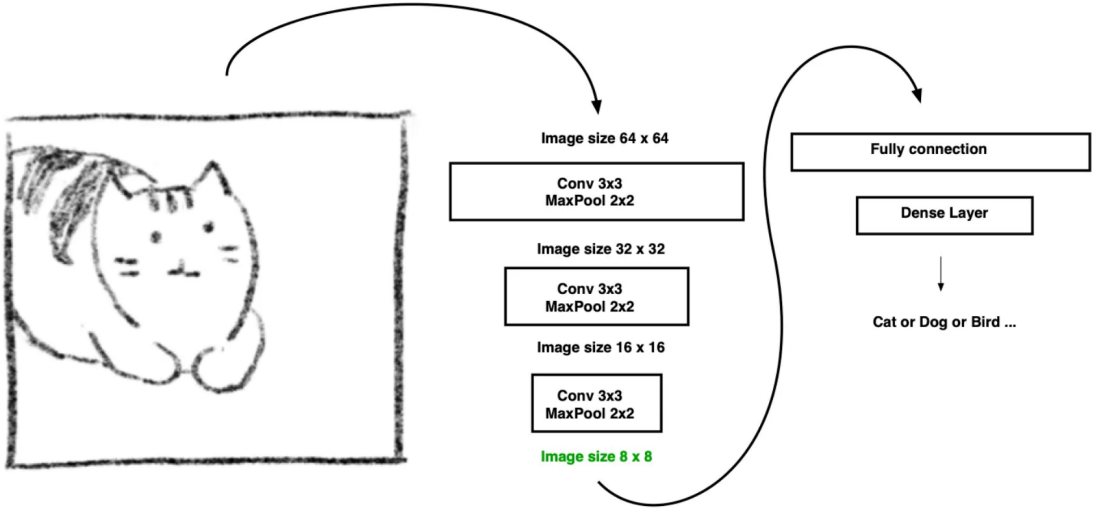


Fig. 8. Full image object classification (we know there is an object)

Assume our input image is 64x64 pixel, then there has an image layer (above figure green text) it's 8x8 pixel image. (we don't care channels here)

- That's what we want, like the DSLR focus points (probability points) to tell us which "pixel" has an object detected.
- Typically any RGB image has three channels (Red, Green, Blue); we are now outputting a channel as the probability.

- Now we try to find parts of an object, the "focal points", the seed for a ROI bounding box proposal
- The image is statically segmented, and each segment is a feature indicator for an object proposal "something is there"

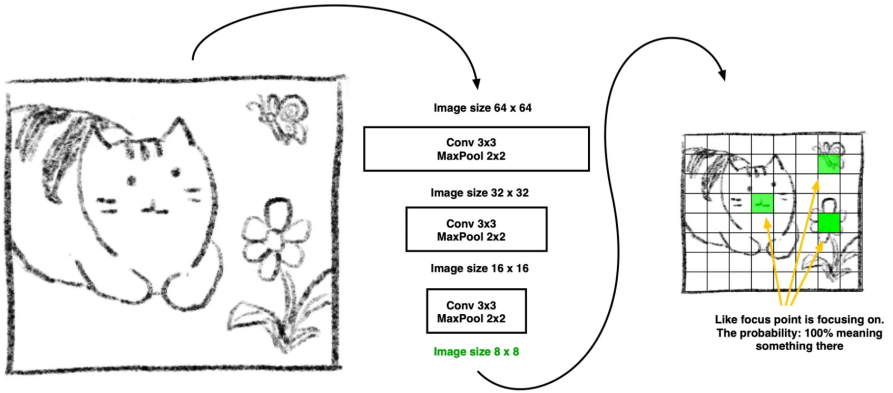


Fig. 9. Static segmentation of an image in equally sized patched; each segment is an object feature detector (there is something nearby)

About the bounding box and the classification.

- As shown above, we can use one channel image as the probability of the object's existence.
- And how about the bounding box? As we can know, a standard bounding box will at least have four numbers (x1, y1, x2, y2, or some people are using x1, x2, width, height, doesn't matter.)
- Now each segment consists of four channels (or each segment outputs a four dimensional vector): The ROI bounding box $\{x_1, y_1, x_2, y_2\}$

- Let's do the same thing as the probability of object exists; we use four channels to represent x_1, y_1, x_2, y_2.

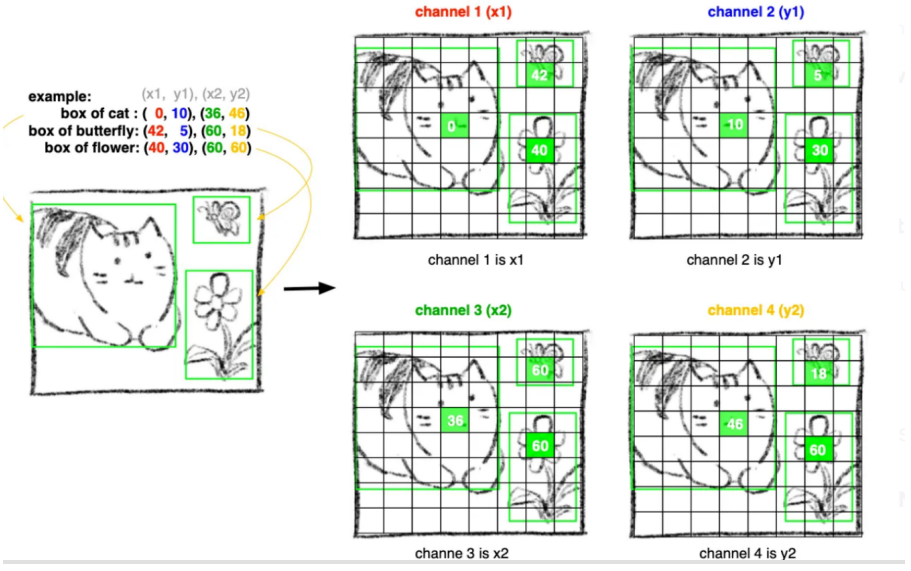


Fig. 10. Region proposal by different output channels of each segment

- We have five channels. For now channels are

$$[\text{probability}, x_1, y_1, x_2, y_2]$$

- If we want to classify objects, say $c \in C=\{c_1,..,c_n\}$, we need and output vector with 5+n elements (or 5+n channels):

$$[\text{probability}, x_1, y_1, x_2, y_2, p_1, p_2, . . , p_n]$$

with p_i as the probability (or better score) that the ROI contains an object (or part of it) of class c_i

- Now we add more channels to enable the object classification, one channel for each class to be predicted and distinguished within a ROI
- Each segment is an object detector, localizer, and classifier (for neighbouring objects)

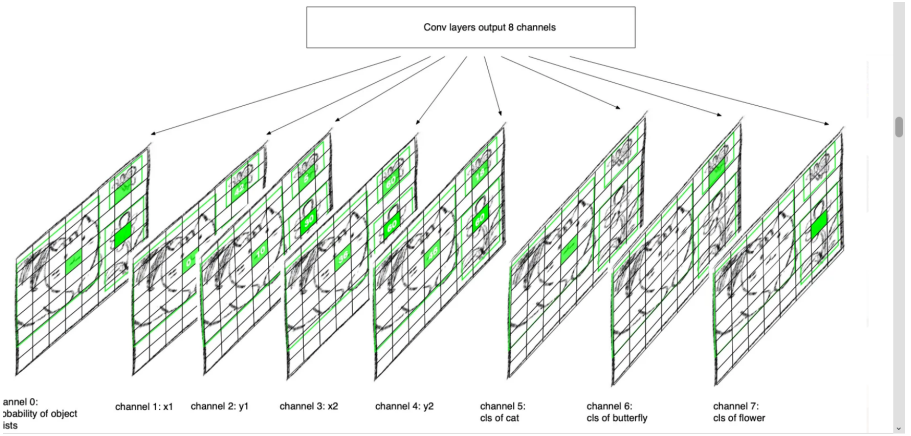


Fig. 11. Region prediction and object classification in segment (anchor) by more output channels for each segment

All we need is a CNN with input shape (width,height,depth) of the input image, and an output "image" of shape (q,p,5+n), with q and p as the 2-dim. shape.

- The entire number of output "pixels" determine the granularity, accuracy, and maximal number of objects to be classified in one input image.
- But we don’t want backward propagation or wrong object output that doesn’t exist (false positive).
  - So we do something like “gate” for bounding boxes channels and classification channels.
  - When the probability is less than 0.5, the gate will be 0 otherwise is 1.

As already mentioned, we need different loss function:

- loss_{p} is for probability loss ⇒ binary-crossentropy (BC)
- loss_{bb} is for bounding boxes loss ⇒ mean-squared-error (MSE)
- loss_{cls} is for classification loss ⇒ binary-crossentropy (BC).

Object parts and fragmentation

- Up to here we assumed that each segment classifier can be recognize one entire object.
- Often, an object is covered by another object, and the "backgrounnd" object can be detected by different fragments as an output form different segments
- Post-processing should try to merge fragments

Encoder-Decoder Networks

- Typically symmetric networks consisting of two parts:
  - An Encoder that reduces the input volume dimension and size (extracting relevant features from the input data) ⇒ Reduction
  - A Decoder that expands the compressed intermediate feature vector to the original input dimension and size (or similar)

Region-proposal CNN (R-CNN)

[Khan]

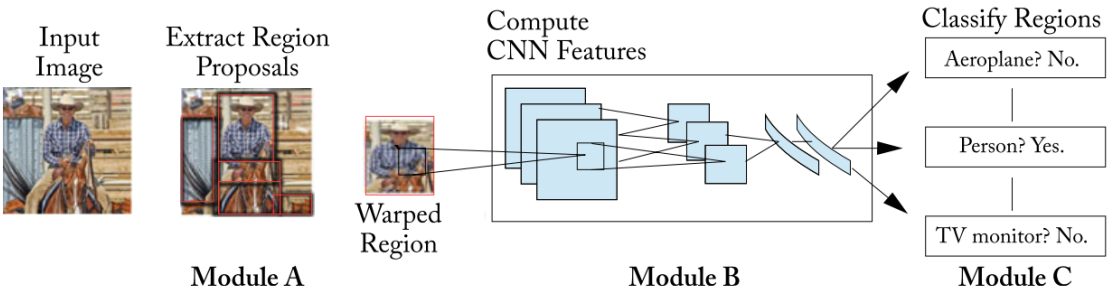


Fig. 12. RCNN object detection system. Input to R-CNN is an RGB image. It then extracts region proposals (Module A), computes features for each proposal using a deep CNN, e.g., AlexNet (Module B), and then classifies each region using class-specific linear SVMs (Module C).

- Given an image, the first module (Module A) uses selective search [Uijlings et al., 2013] to generate category-independent region proposals, which represent the set of candidate detections available to the object detector.
- The second module (Module B) is a deep CNN (e.g., AlexNet or VGGnet), which is used to extract a fixed-length feature vector from each region.
- In both cases (AlexNet or VGGnet), the feature vectors are high-dimensional.
- In order to extract features from a region of a given image, the region is first converted to make it compatible with the network input. More precisely, irrespective of the candidate region's aspect ratio or size, all pixels are converted to the required size by warping them in a tight bounding box.

- Next, features are computed by forward propagating a mean-subtracted RGB image through the network and reading off the output values by the last fully connected layer just before the soft-max classifier
- After feature extraction, one linear SVM per class is learned, which is the third module(C) of this detection system.
- Note that selective search produces many region proposals Multiple stages must trained independently
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- R-CNN runtime roughly 47 seconds per image (even using GPUs)

Loss Functions

- Multi-task training with 4 loss functions (obj/not obj, ROI bbox, classify, final obj bbox)

[Brown, 2017]

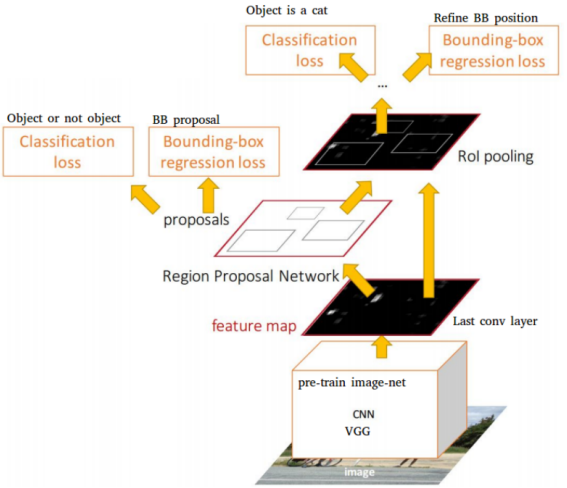


Fig. 13. REgion-proposal and classification networks uses different loss (erro) functions for different features

Quality Assessment and Metrics

- Assessing the quality of a classification result is generally well defined
- Quality assessment of object localization and segmentation results is more complex
- Object localization output is a bounding box
  - How to assess overlap between ground truth and computed bounding boxes?
  - What about sloppy or loose ground truth bounding boxes?
- Segmentation output is polygon-like pixel region
  - How to assess overlap of polygon-like ground truth and computed output region?
  - What about sloppy or corse ground truth regions?

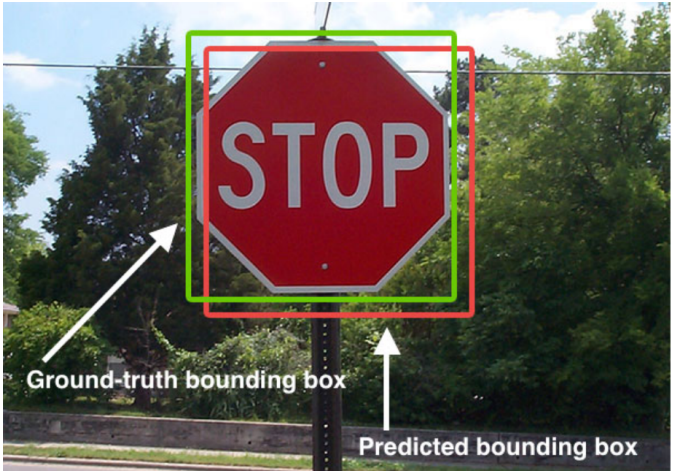


Fig. 14. Ground-truth and intersection over union (IoU)

Intersection over union (IoU)

- Each bounding box (i.e. detection) is associated with a confidence (sometimes called rank)
- Detections are assigned to ground truth objects and judged to be true/false positives by measuring overlap
- To be considered a correct detection (i.e. true positive), the area of overlap a_{ovl} between predicted bounding box BB_{p} and the ground truth bounding box BB_{gt} (training label) must exceed 0.5 according to:

$$a_{ovl} = \frac{BB_p \cap BB_{gt}}{BB_p \cup BB_{gt}}$$

- a_{ovl} is often called intersection over union (IoU)

[Brown, 2017]

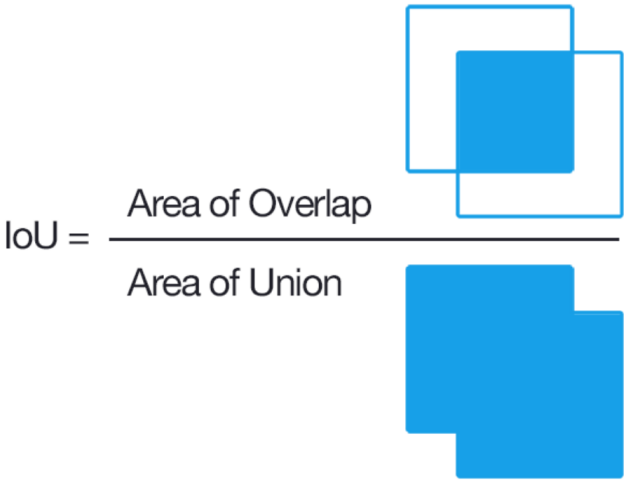


Fig. 15. Intersection over unoin illustration

[Brown, 2017]

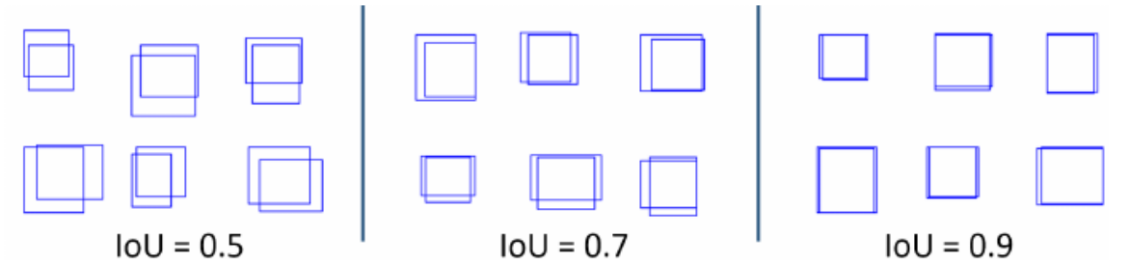


Fig. 16. Intersection over unoin examples

Common Issues with Algorithms

- Compute performance often poor
  - Too many region proposals to test and label
  - Difficult to scale to larger image size and/or frame rate
  - Cascading approaches help but not solve
  - Aggressive region proposal suppression leads to accuracy issues
- Accuracy problems
  - Huge number of candidate regions inflates false-positive rates
  - Illumination, occlusion, etc. can confuse test and label process
- Not really scale invariant
  - Early datasets not very large so limited feature variation
  - Now training datasets are many TB – helps but doesn’t solve
  - Large variation of feature scale can inflate false-negative rates

Anomaly Detectors

- An anomaly detector is trained only with ground-truth base-line examples, i.e., without any of the objects to be detected later
- An anomaly detector is basically a binary classificator: There is nothing / There is something.
- Commonly, auto-encoder (encoder-decoder) AE architectures are used, and trained in such a way that the network should output the input (image).
- The network learns the "background", e.g., micrograph images with a specific texture, but without cracks
- It learns to reconstruct the original input data by relevant information, e.g., the texture consists of a cross lien pattern, and only some geometric features are used to recosntruct the pattern (line distance, angle, intensity)

- The output of an AE is then compared with the input, calculating an mean average error (MAE)
- If the MAE is over a threshold, something "not normal" was found!

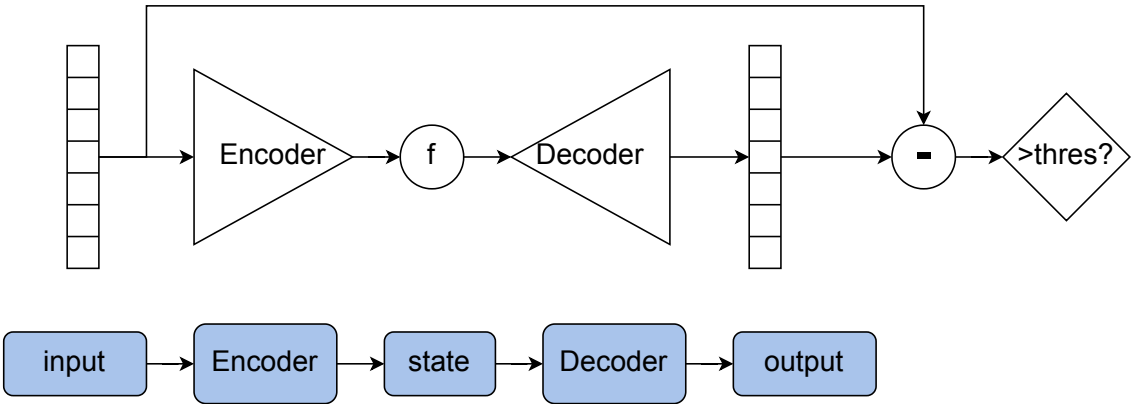


Fig. 17. Basic architeture of an anomaly detector

U-Net

- Typical Encoder-Decoder architecture for pixel segmentation and data augmentation!

There is large consent that successful training of deep networks requires many thousand annotated training samples.

- U-Net is a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently.
- The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization.

Ronneberger et al. showed that such a network can be trained end-to-end from very few images and outperforms the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neu- ronal structures in electron microscopic stacks.

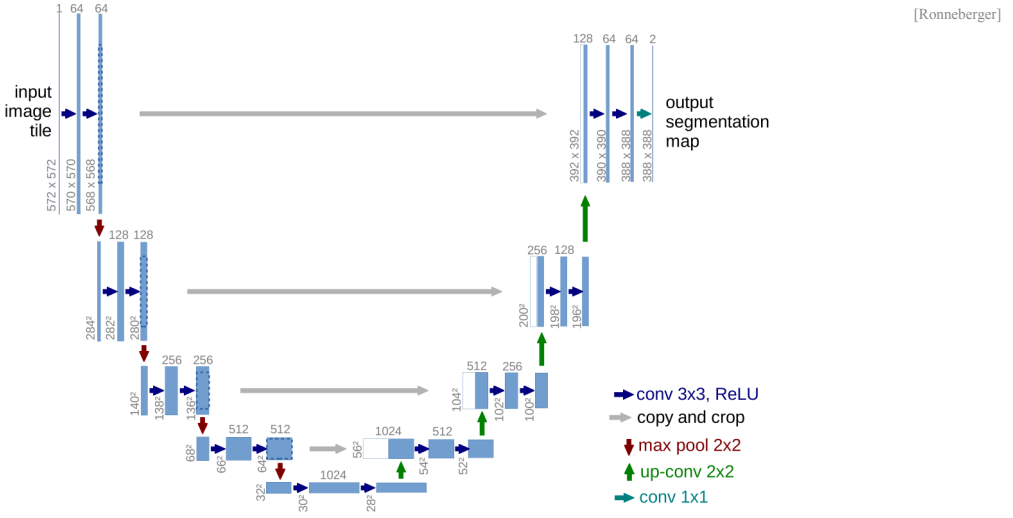


Fig. 18. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Fast R-CNN

- Fast R-CNN combines stages B and C of R-CNN and is trained with multi-task loss (log loss and smooth L1 loss)
- A Fast R-CNN network takes as input an image and a set of object proposals
- The network first processes the whole image with conv and pooling layers to produce a conv feature map.
- The input to Fast R-CNN is an entire image along with object proposals, which are extracted using the selective search algorithm

Fast R-CNN

- For each object proposal an region of interest ROI pooling layer extracts associated features from the conv map, i.e., a feature vector of fixed size is then extracted from the feature maps by a Region of Interest (RoI) pooling layer (Module B).
- The role of the RoI pooling layer is to convert the features, in a valid RoI, into small feature maps of fixed size (X □ Y , e.g., 7 □ 7), using max-pooling.
  - X and Y are the layer hyper-parameters.
- A RoI itself is a rectangular window that is characterized by a 4-tuple that defines its top-left corner (a, b) and its height and width (x,y).

Fast R-CNN

- Fast R-CNN is still using an independent region proposal stage.
- Each feature vector is then given as input to fully connected neural layers, which branch into two sibling output layers.
  - One of these sibling layers (Module C) gives estimates of the soft-max probability over object classes and a background class.
  - The other layer (Module D) produces four values, which redefine bounding box positions, for each of the object classes.

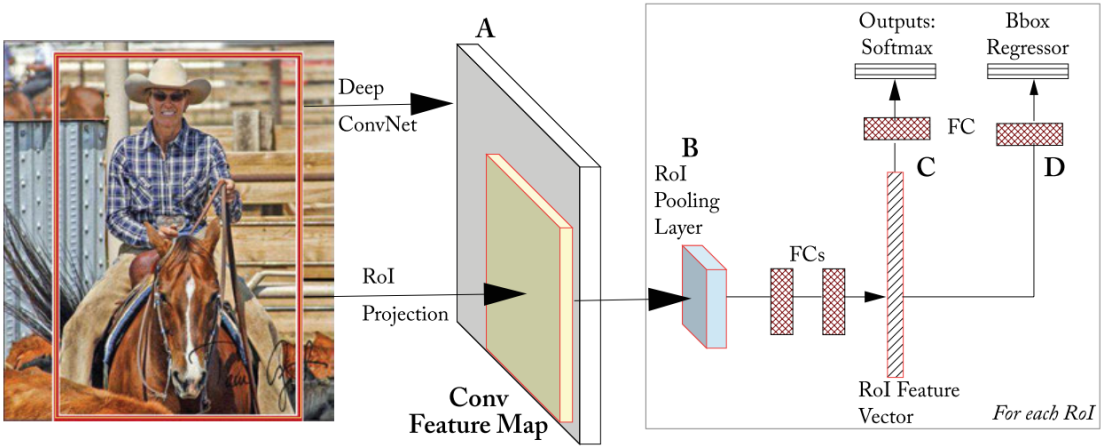


Fig. 19. Fast R-CNN architecture and data flow

Faster R-CNN

- Faster R-CNN revision will introduce a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network.
- The RPN simultaneously predicts object bounds and objectness scores at each position.
- The RPN is trained end-to-end to generate high-quality region proposals which are used by Fast R-CNN
- The RPN and Fast R-CNN are merged into a single network by sharing their convolutional feature. Using “attention” mechanisms, the RPN component tells unified network where to look.
- Faster R-CNN generates about 300 proposals per image

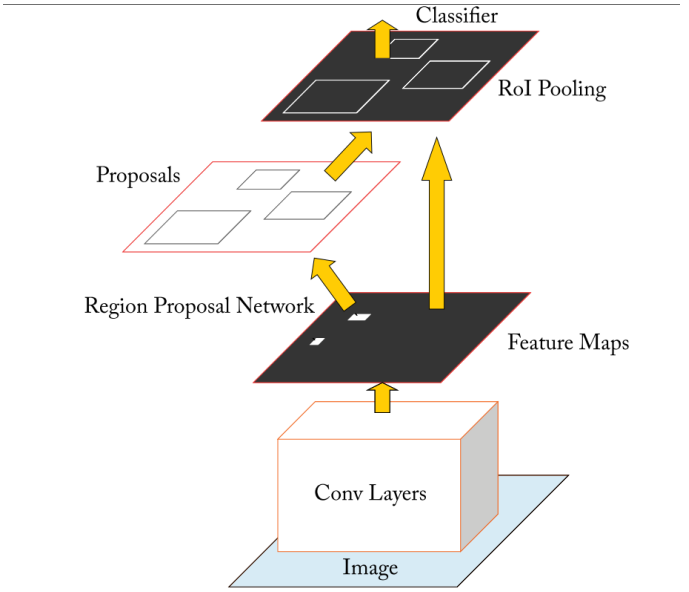


Fig. 20. Faster R-CNN: Combining RPN with fast R-CNN object detector

Semantic Segmentation

- Typical classification networks take fixed-size images as input and produce non-spatial output maps, that are fed to a soft-max layer to perform classification.
- The spatial information is lost, because these networks use fixed dimension fully connected layers in their architecture.

Pixel Classifier

- Pixel classifier: Central pixel classification by neighbourood (high spatial accuracy)
- Segment classifier: Segment classification by entire segment pixels (medium spatial accuracy)
- Fully convolutional networks can take (1) input images of any size and produce (2) spatial output maps. These two aspects make the fully convolutional models a natural choice for semantic segmentation.

- Pure pixel classifiers are computational intesive. Each output pixel requires the computation of a CNN with the input segment sub-image.

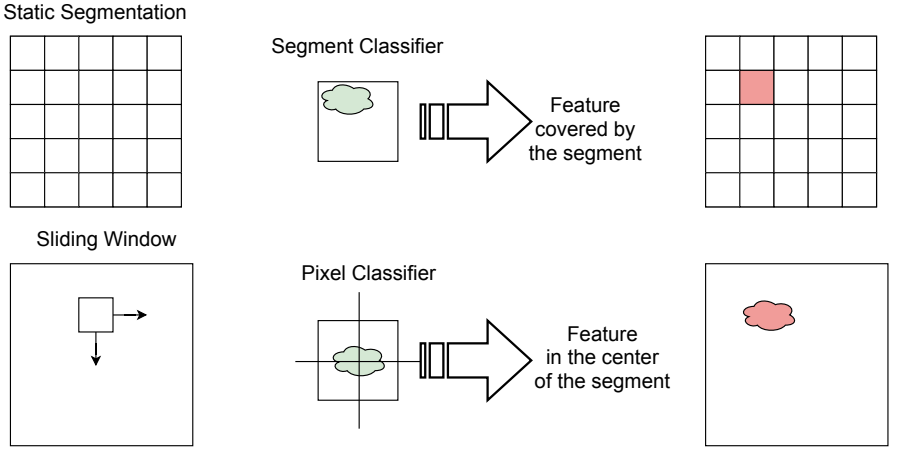


Fig. 21. Difference between static segment classification and pixel classification using a moving and sliding window

Fully Convoultional Network (FCN)

[Khan]

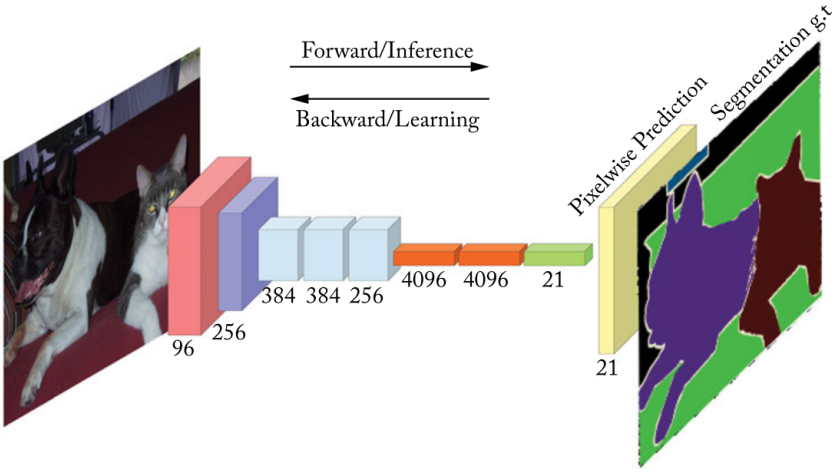


Fig. 22. Pixel classifier with convoultional layers only

Segmentation Examples

[Brown, 2017]

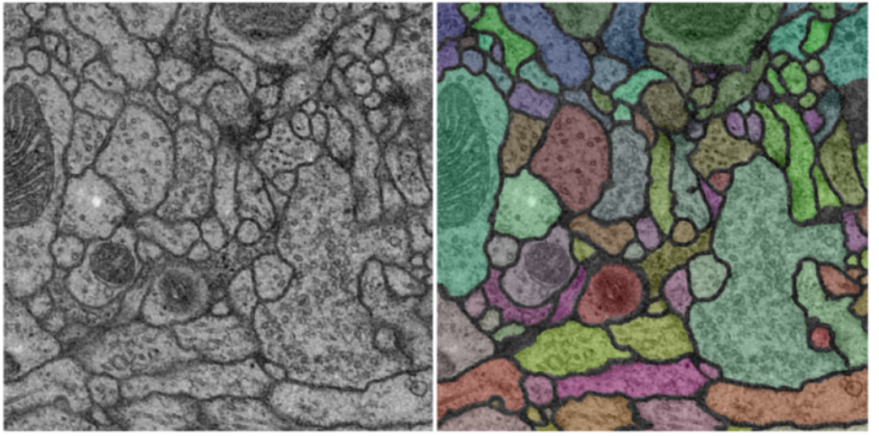


Fig. 23. Ciresan - Neuronal membrane segmentation

[Brown, 2017]

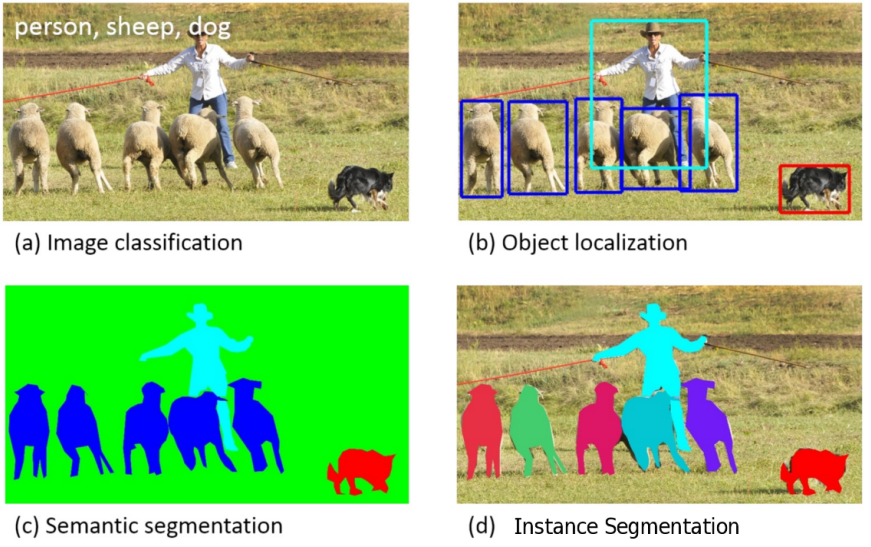


Fig. 24. Different levels of object detection and segmentation

Pre-trained Object Detectors

coco-ssd

COCO: Common Objects in Context, SSD: Single Shot Detector

yolo

Yolo: "You Only Look Once" system, an open-source method of object detection that can recognize objects in images and videos swiftly whereas SSD runs a convolutional network on input image only one time and computes a feature map.

- YOLO can struggle to localize objects properly, but has fewer background errors (fp)

Model-driven Segmentation

- Up to here we only considered data-driven model-less region proposal networks
- But in measuring technologies and measuring data we have mostly an idea about geometric features of ROIs
- Edge detection, e.g., combined with point clustering methods can be used to propose ROIs very fast and accurately

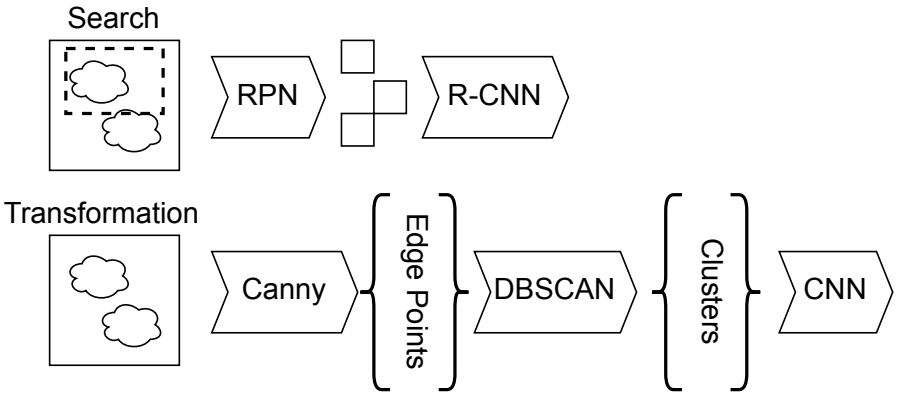


Fig. 25. (Top) Region proposal by search and data-driven learning (Bottom) Model-based images transformation and point clustering

Anomaly Detection in Tomography Data

Supervised CNN

- Three-dimensional data, e.g., from x-ray ct-scans, can bereduced to a set of lower-dimensional data:
  - 2D Image slices indexed along a geometric axis, e.g., the z-axis (depth image stack)
  - 1D signals along a geometric axis, e.g., z-slices $s_z(x,y)$
- CNNs can be applied to arbitrary dimensional data, indeed, n-dimensional data is commonly processed as a linear one-dimensional array (linearily packed multi-dimensional data)

Anomaly Detection in Tomography Data

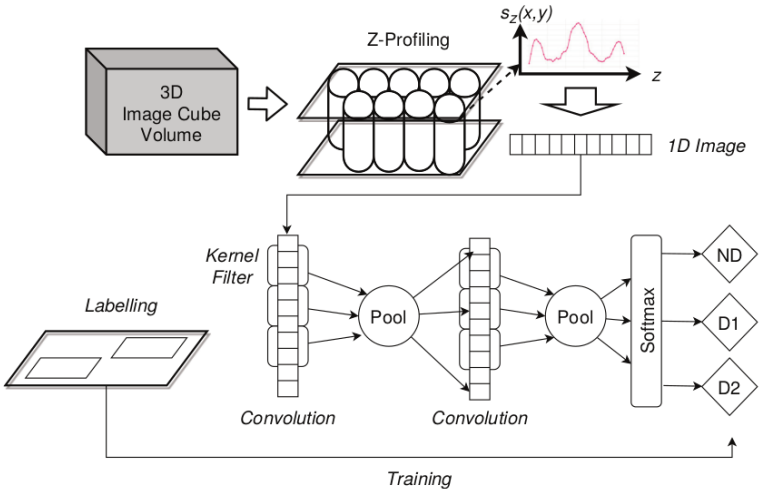


Fig. 26. Z-profile signals as 1D images as input for a CNN damage classifier (ND: No damage class, D1: Damage 1, D2: Damage 2, and so on) ⇒ Pixel sgementation detector!

Anomaly Detection in Tomography Data

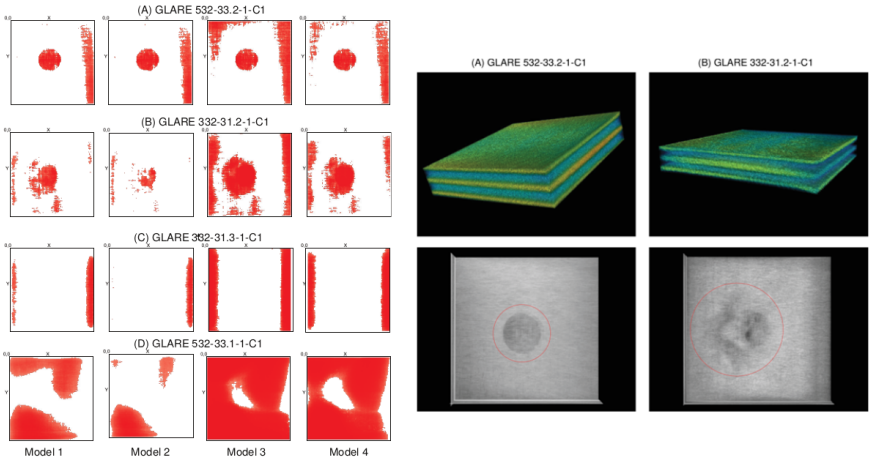


Fig. 27. (Left) Damage feature maps retrieved from four different CNN classifiers and for the specimen A (training and prediction), B, C, and D) (Right) CT image volume and selected x-y slice visualization (A-B) With centred resin defect in the PREG layer)