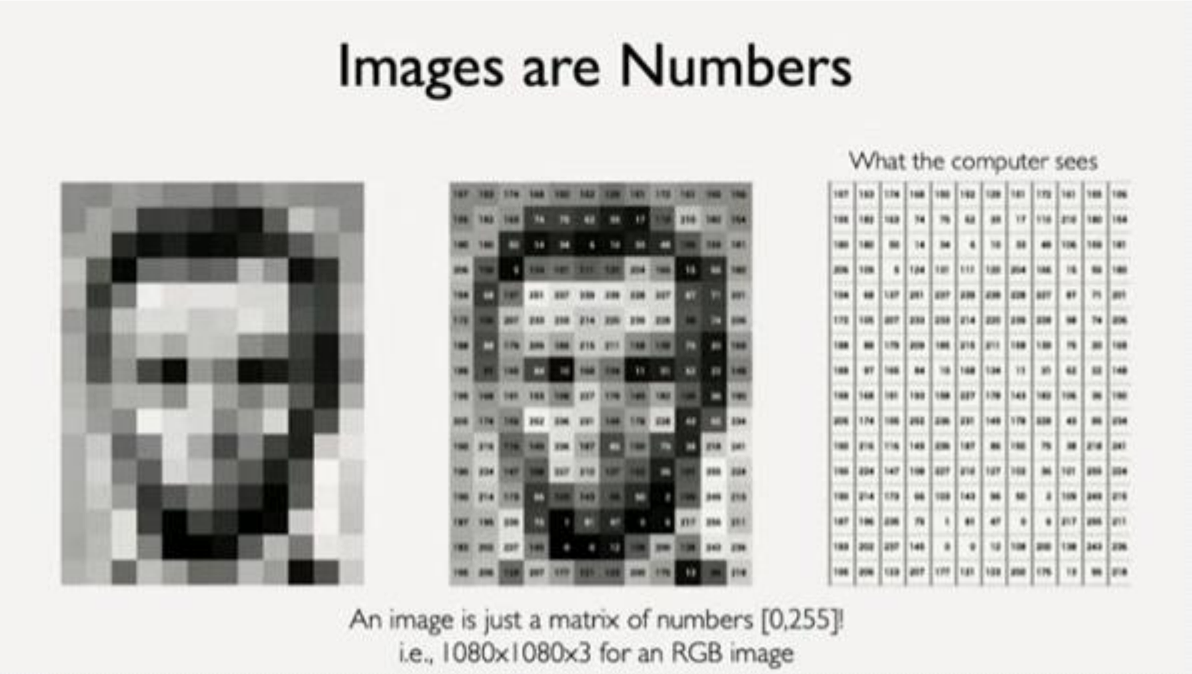
Intro to deep learning 3

Wednesday, August 20, 2025 12:46 PM

Lecture 3: Convolutional neural network - deep computer vision

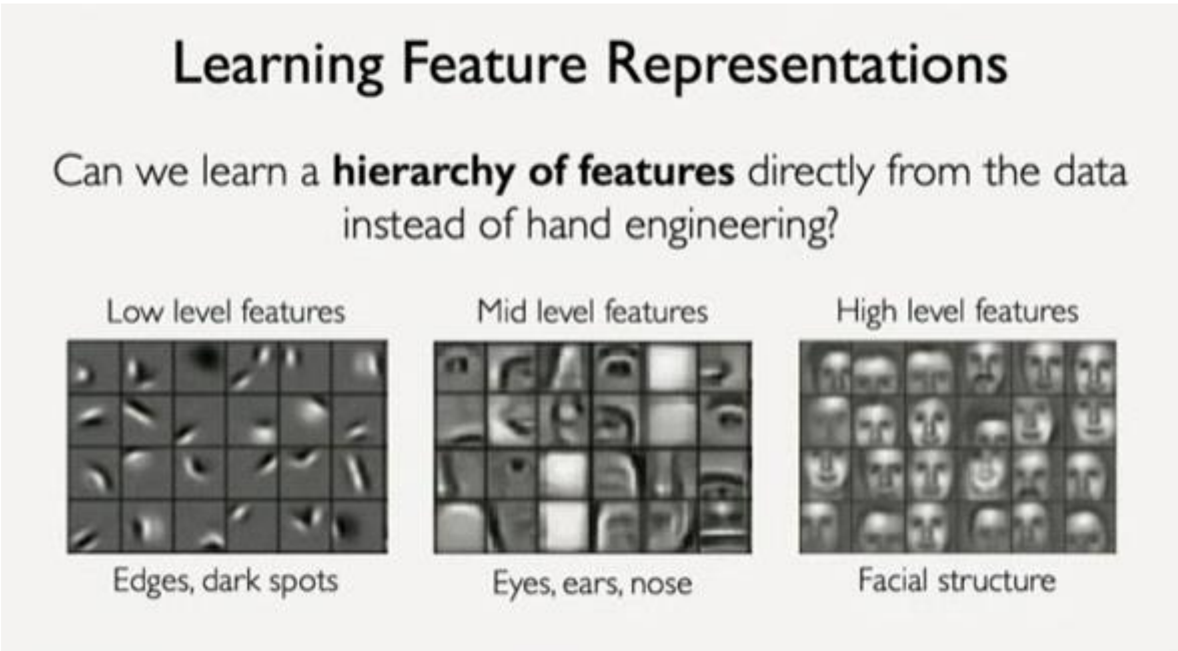
How can a computer process an image? Images are simply array of numbers in 2D (times 3 for colors). We will focus on **regression** (to output a continuous value) and **classification** (to output a class label through a set of probabilities) based on images input.



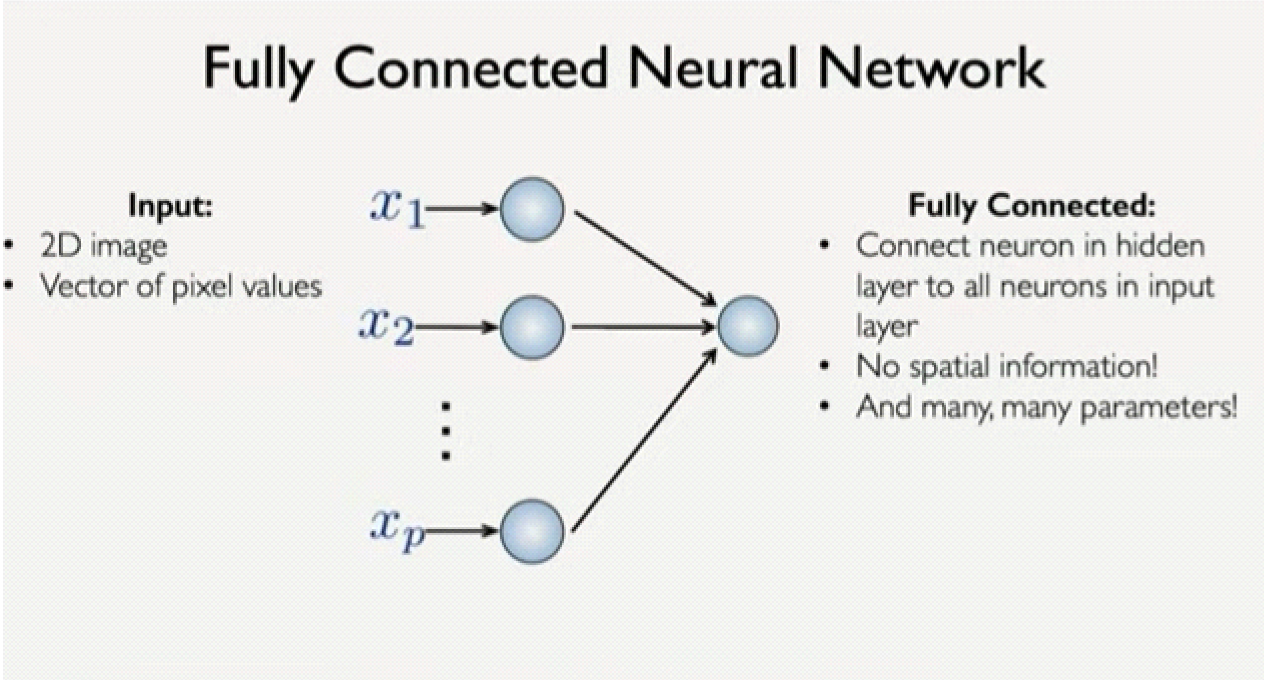
We need models able to distinguish unique features. This is the task of **features detection**

Defining features by hand is basically impossible because of variations in features, angle of view, … Our models need to be invariant to those modifications in the feature space.

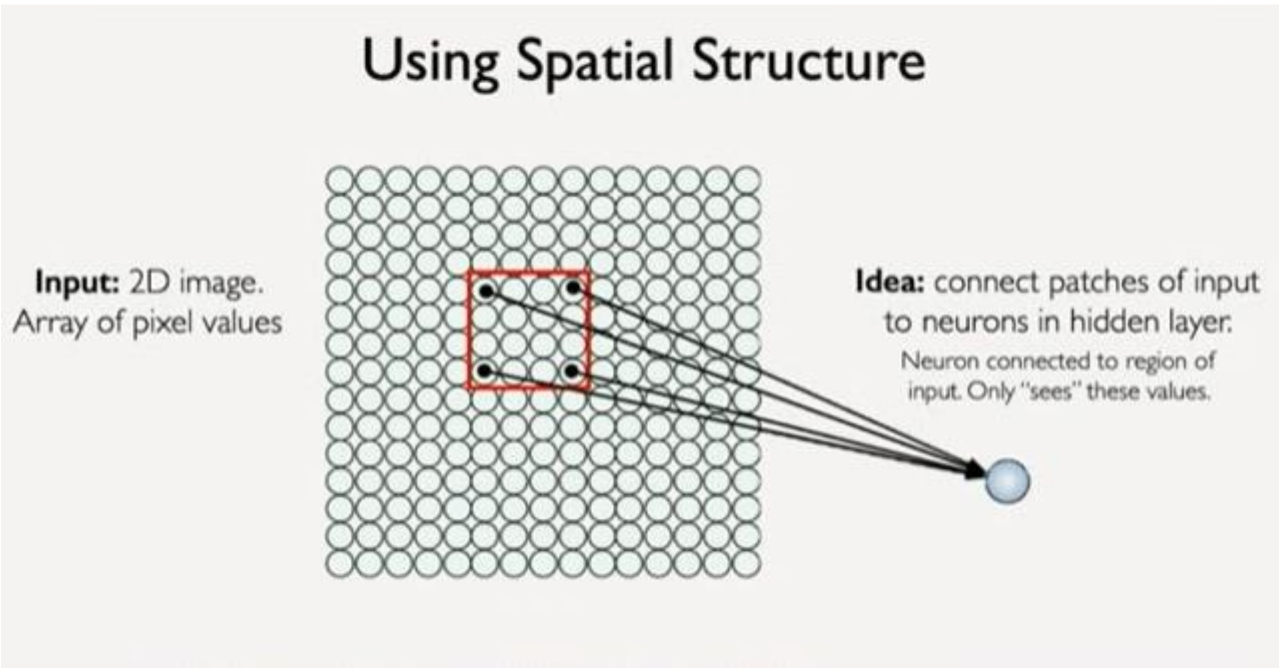
ML tries to define the features while DL tries to detect them starting from data to defining the features implicitly and not manually

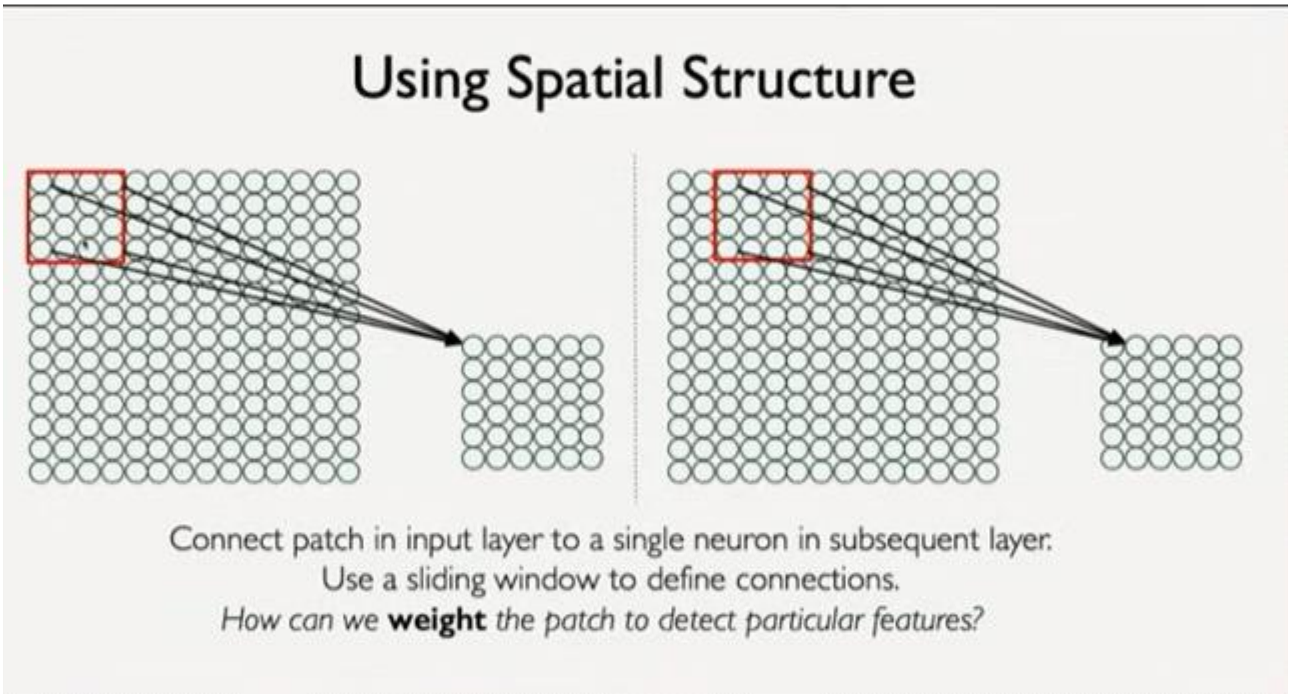


If we wanted to use the fully connected NN from lecture 1, we can do it as follows. But we have problems. For example the 2D input must be flattened and the spatial information is lost. Because of the full connection between layers, there is also a LOT of parameters

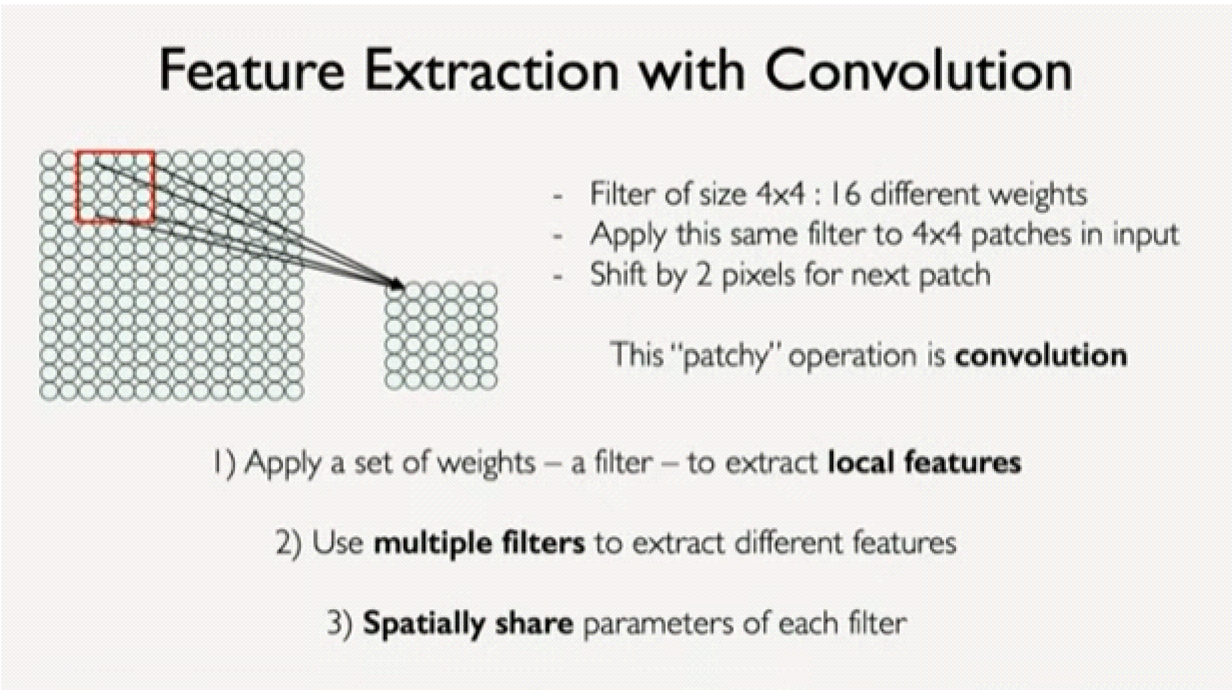


We want instead to preserve spatial information native to the data to inform the architecture. We represent the 2D image as 2D array of pixels. We connect patches of pixels to neurons in a sliding fashion.

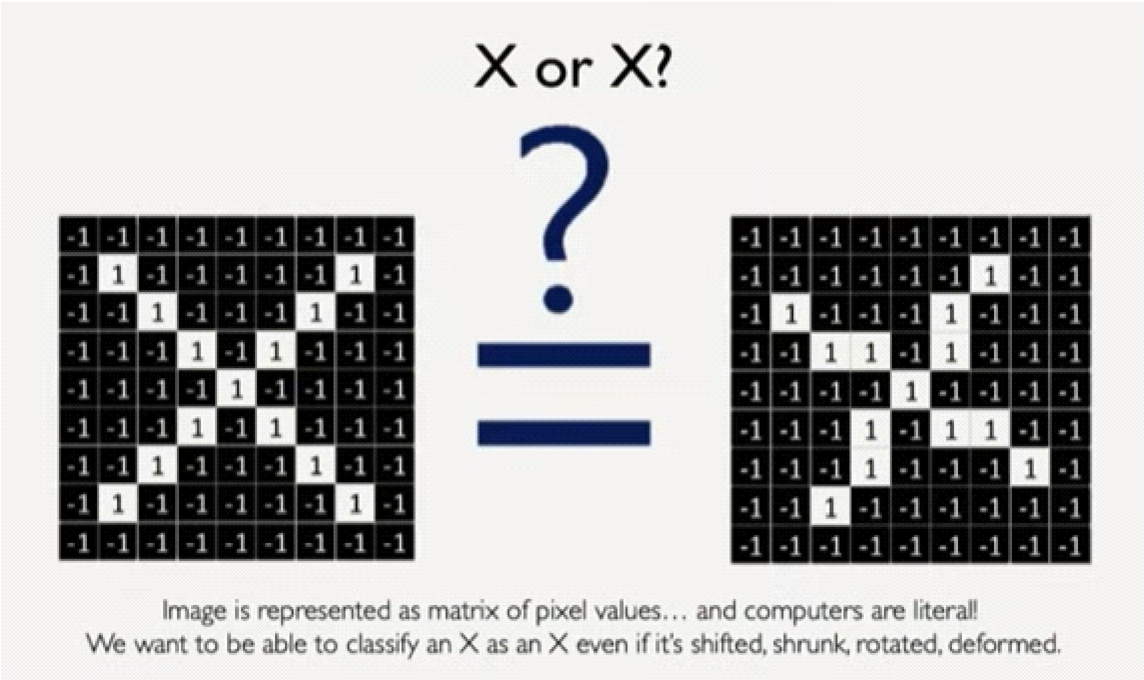




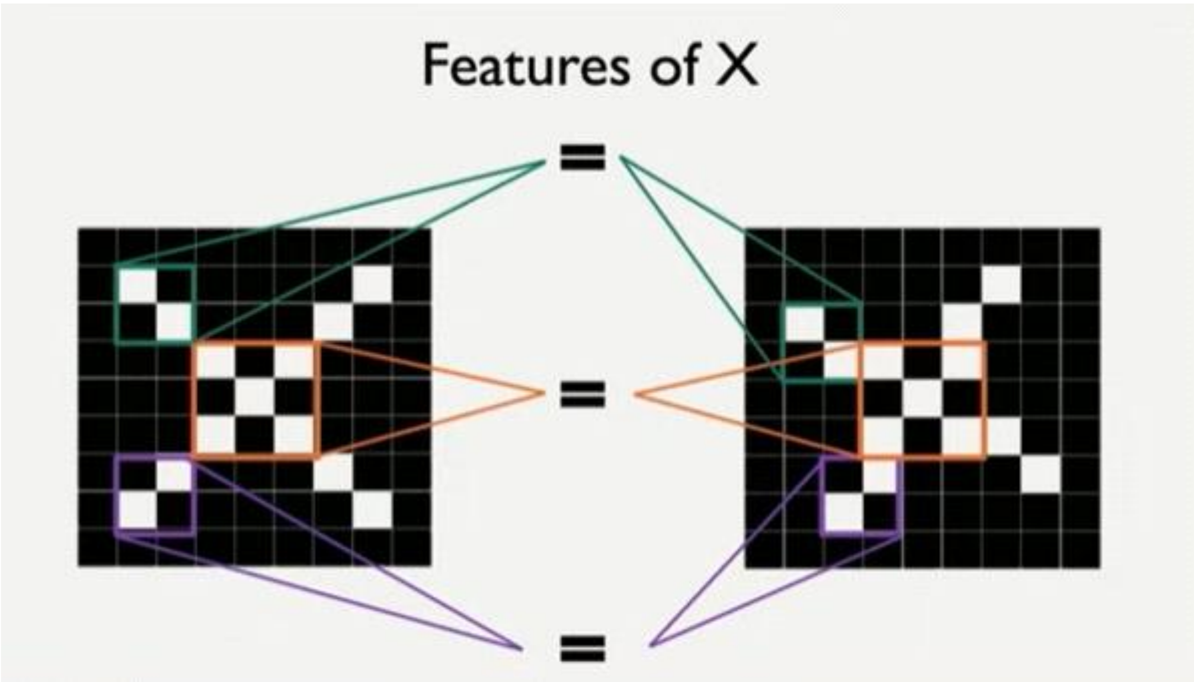
Features are extracted with convolution and each patch is multiplied pointwise by a patch-size set of weights called "filter" and then sum everything.

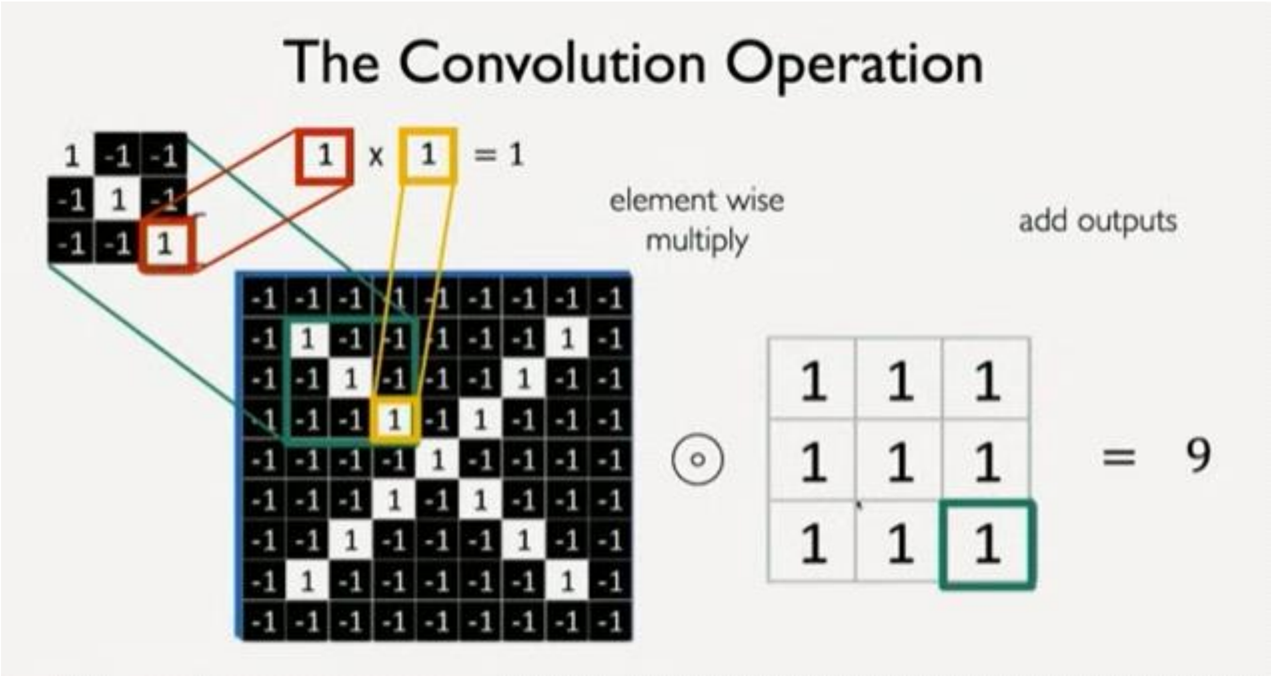


How is this convolution able to define the feature information?
Example:

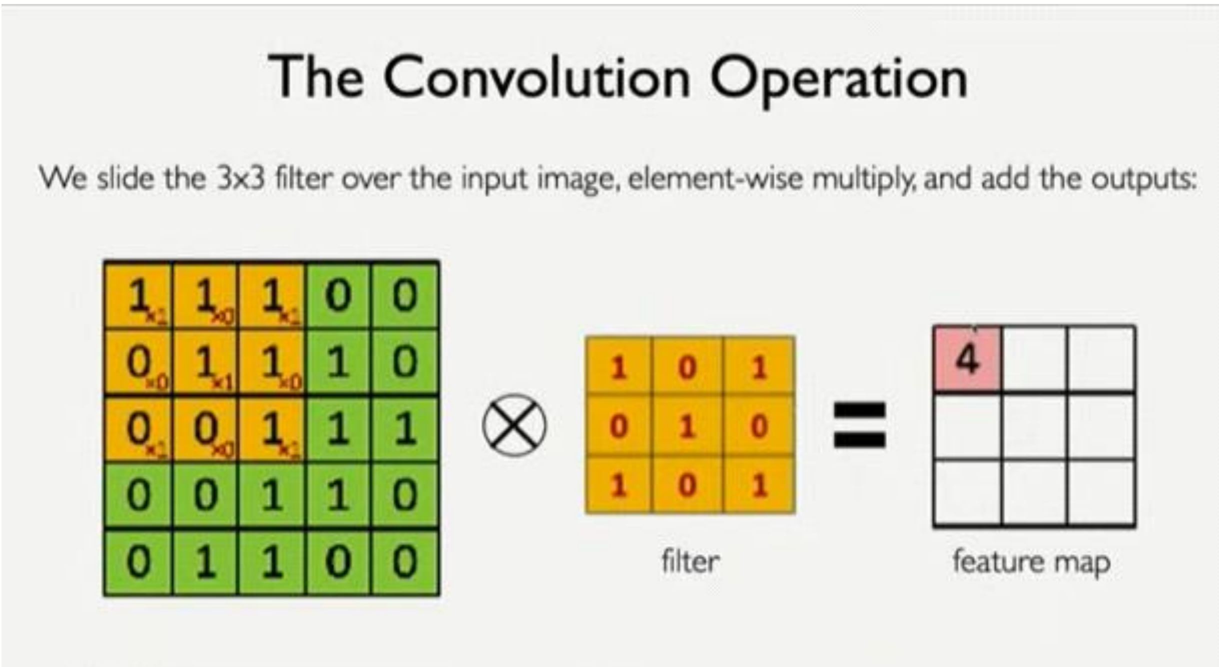


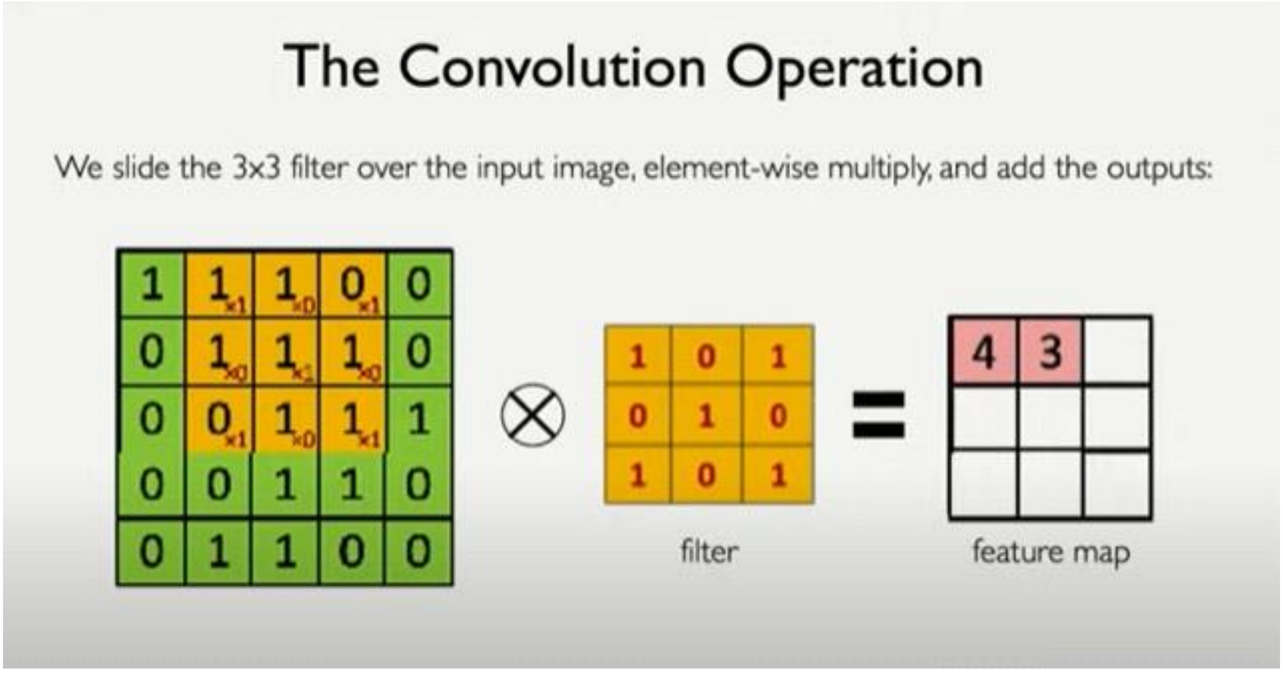
A patchwise comparison seems already a better alternative than just looking at the whole picture for comparison

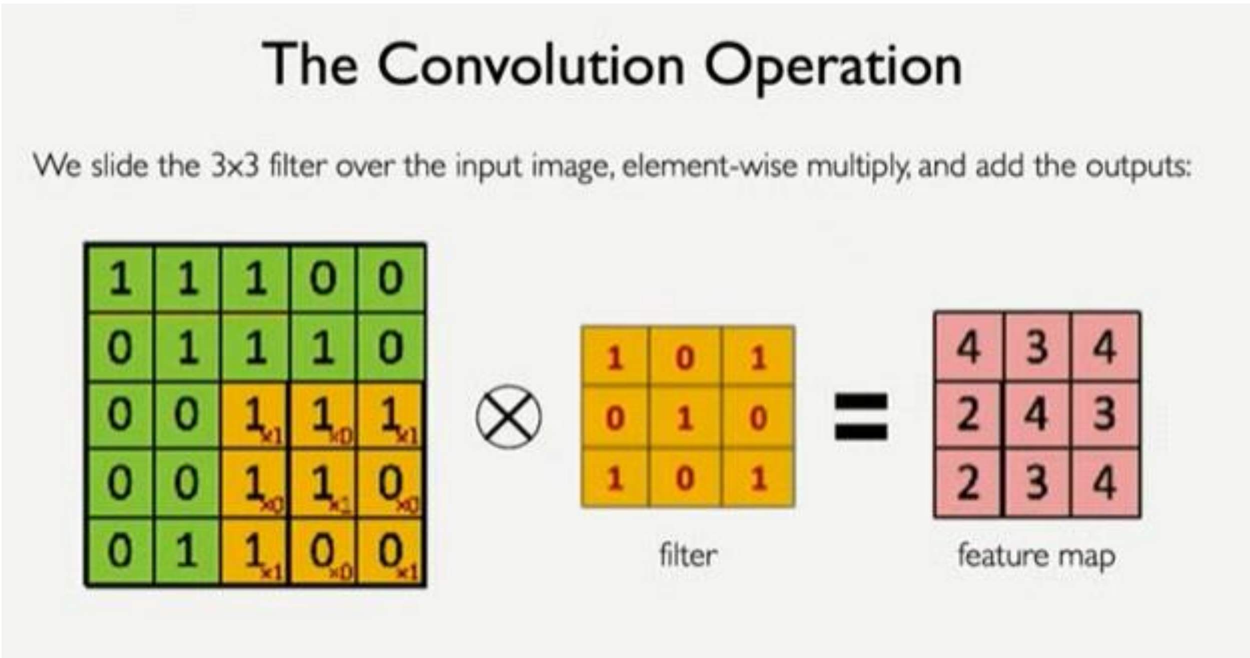




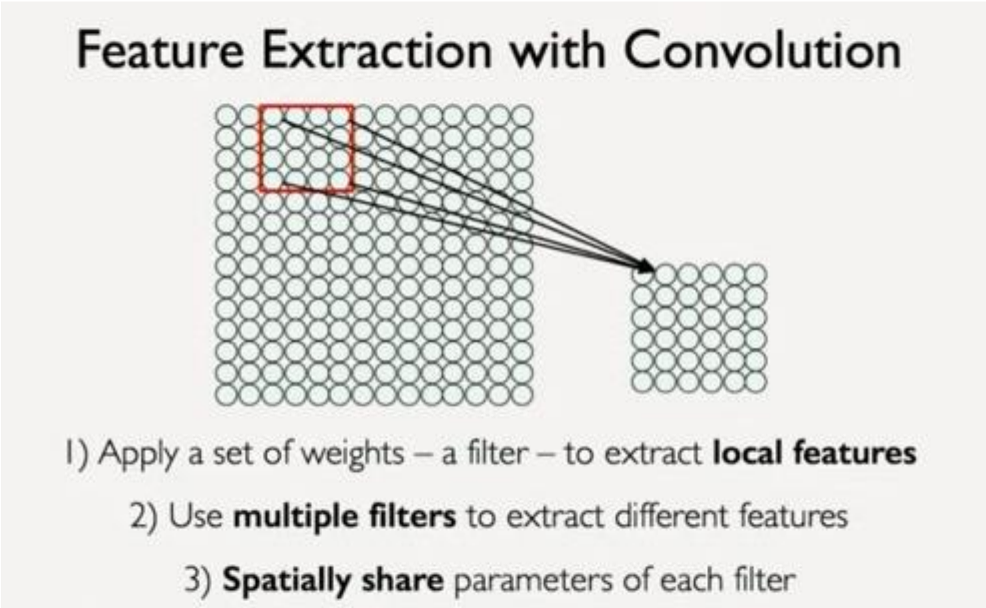
Another example of convolution for fixing the idea





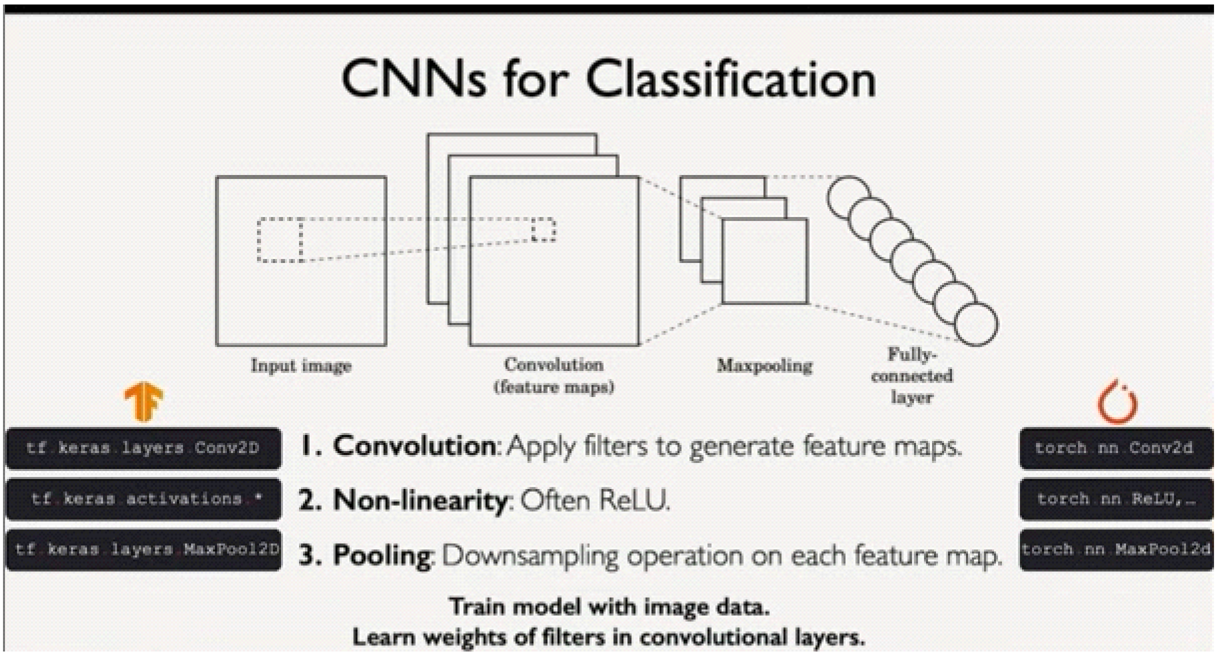


Which filter you choose is fundamental for the extraction of features

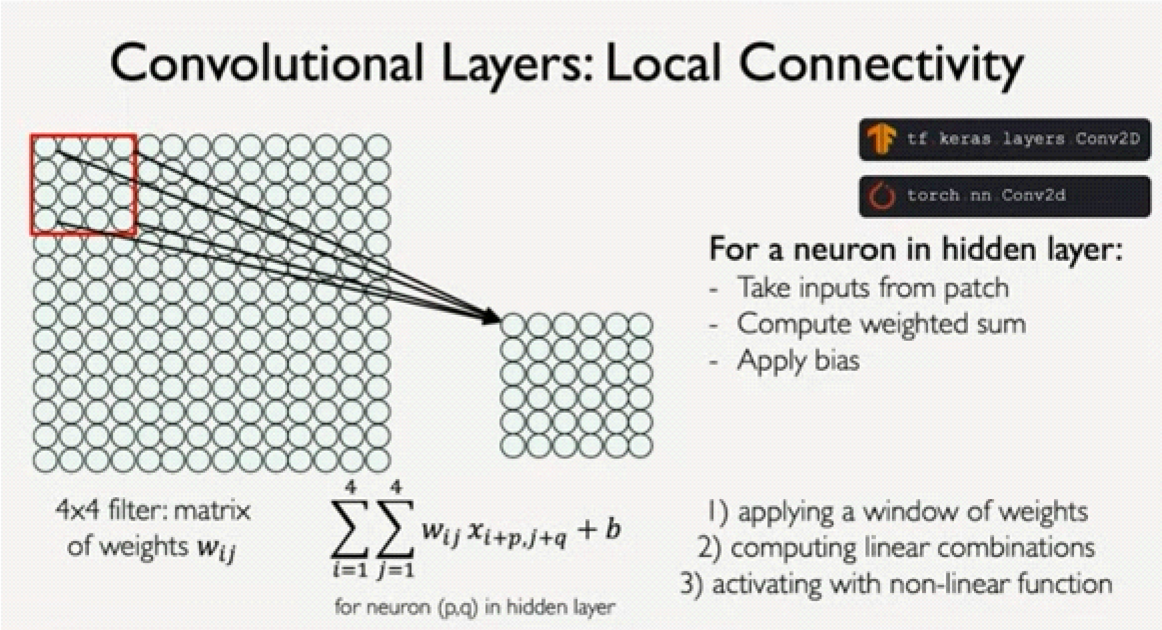


Convolutional Neural Networks

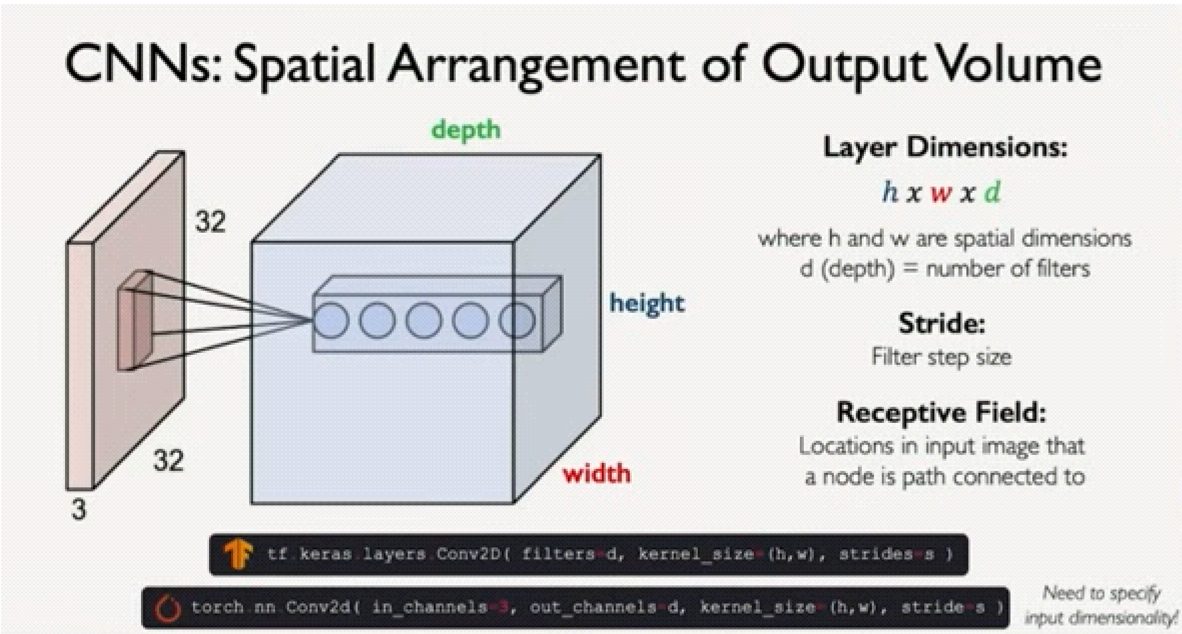
In CNNs we have the convolutions happening on the patches of an image with the filter. Then we do pooling downsampling the feature maps from convolutions.



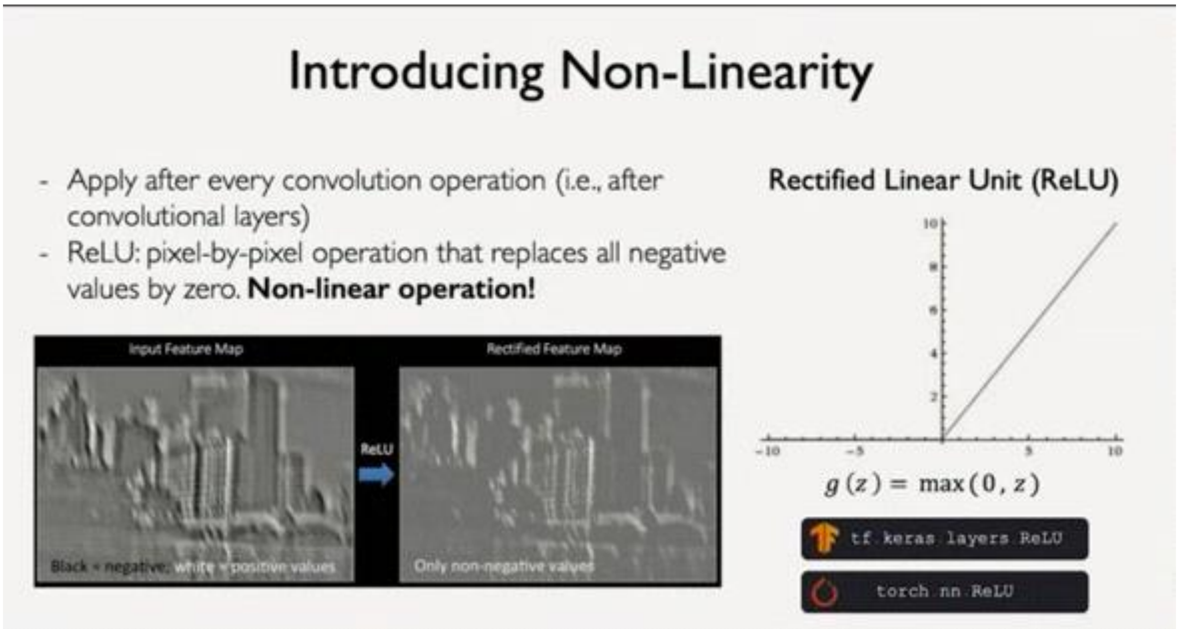
More in depth in the convolution note each pixel sees only its patch in the operation.



Now, when you do convolution, you do not have only ONE filter, but you have multiple ones, so the feature extraction gives a VOLUME of features.



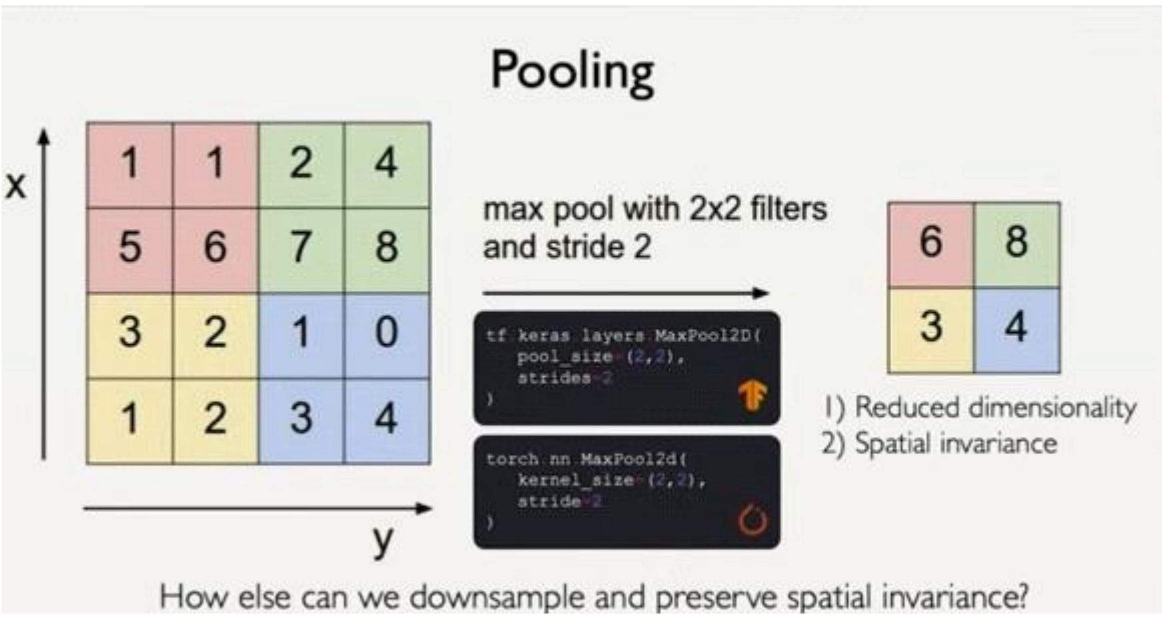
After convolution we need to apply a ReLU (linear rectifier) which puts to 0 all negative values. It works like a threshold and is used in computer vision. Remember it is applied on the feature maps.



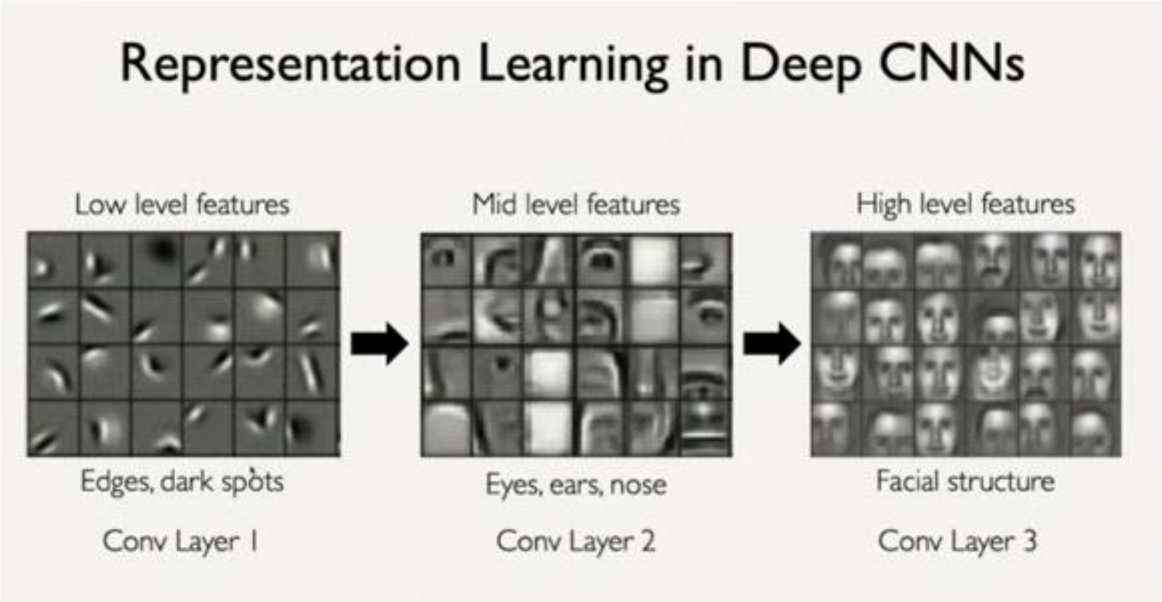
Note from questions: pooling is used because you stack multiple convolutions and you are able to detect features at different scales, which you then combine when you apply your network to an input after training.

Pooling

Often pooling is done with max pooling taking the max in 2x2 filters because it can change a gradient a lot in optimization (mean pooling would make very stable gradients).

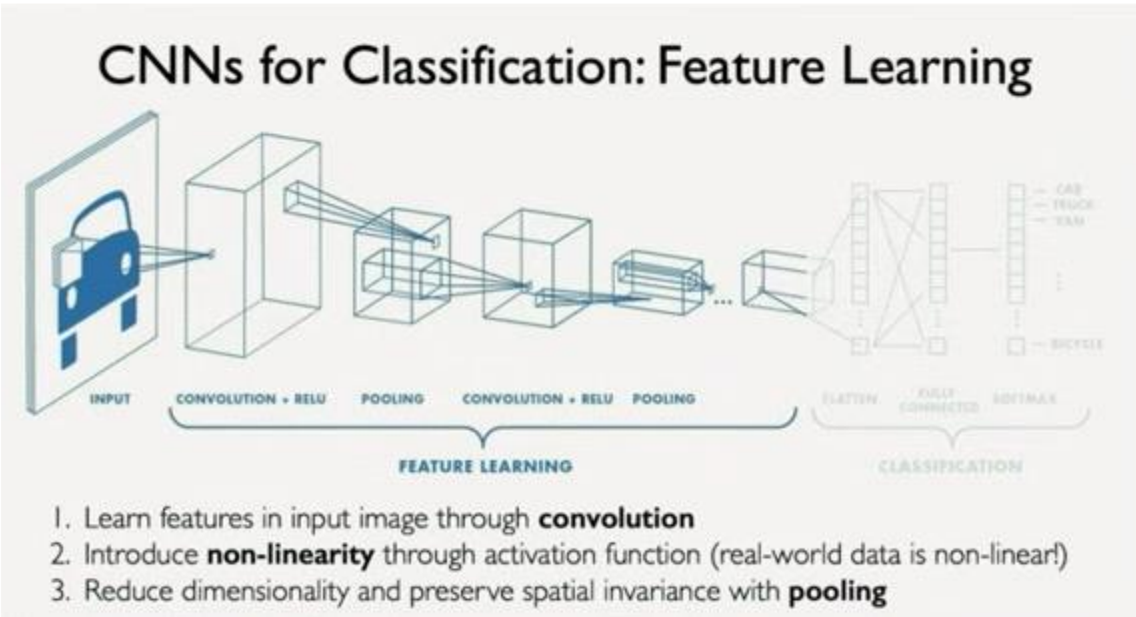


In the end you learn the features without defining them and those features will act at different scales.

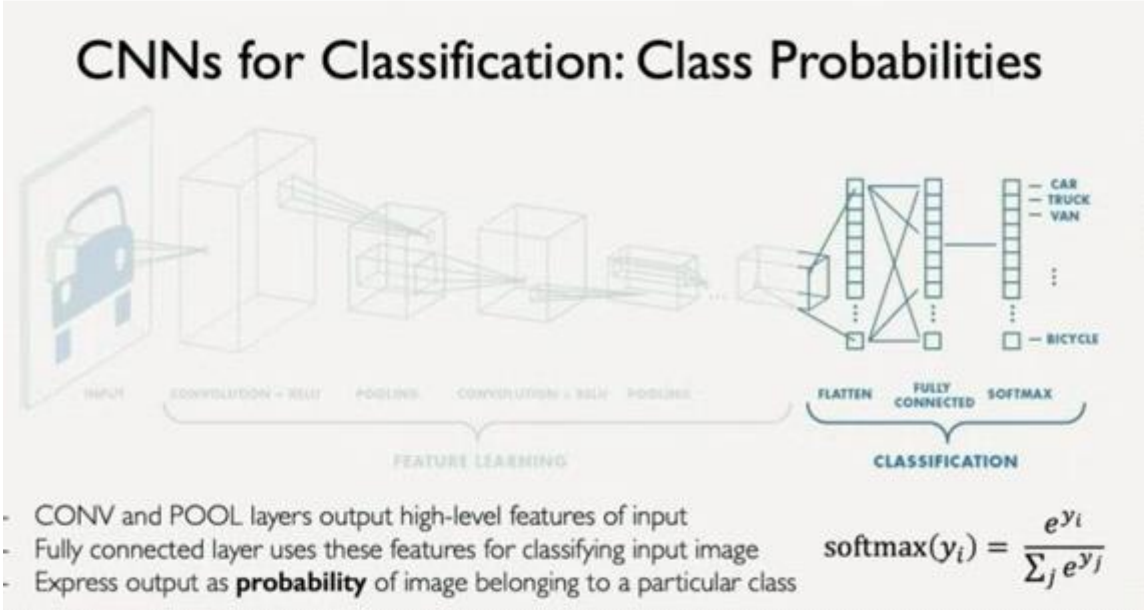


Classification

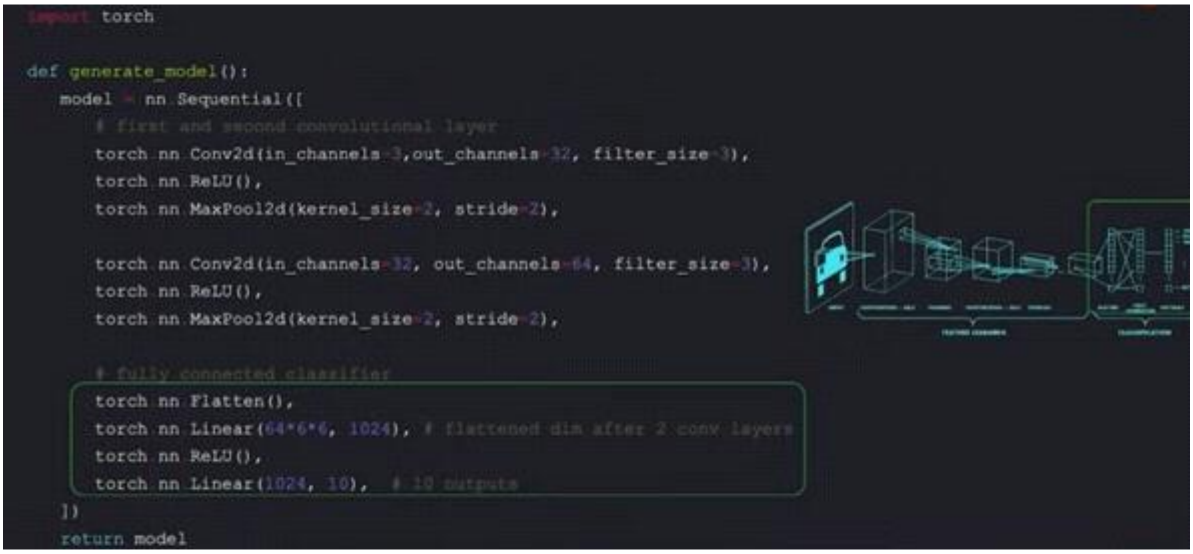
Now we handled the feature learning



We need to then classify. We use a softmax function to create a probability distribution on the flatten output of the feature extrections going through fully connected layers.

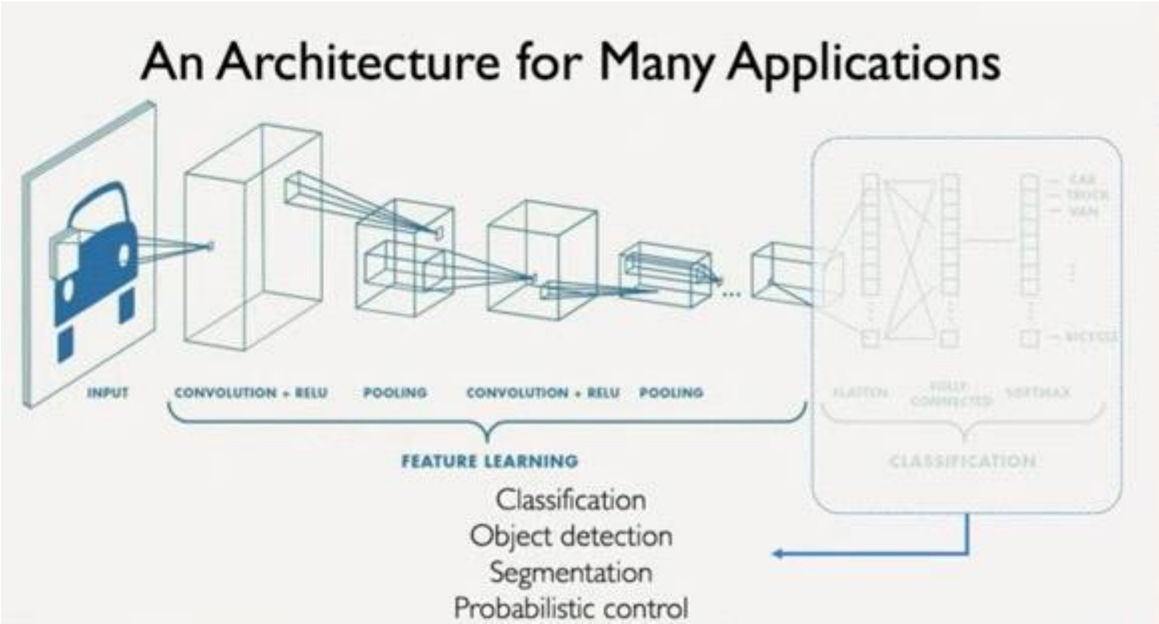


How to choose features? Start quite small, but upsample through layers). Also, depending on image size, some resolutions might not make sense for too few pixels, so it depends on the image size, but also on the application type.

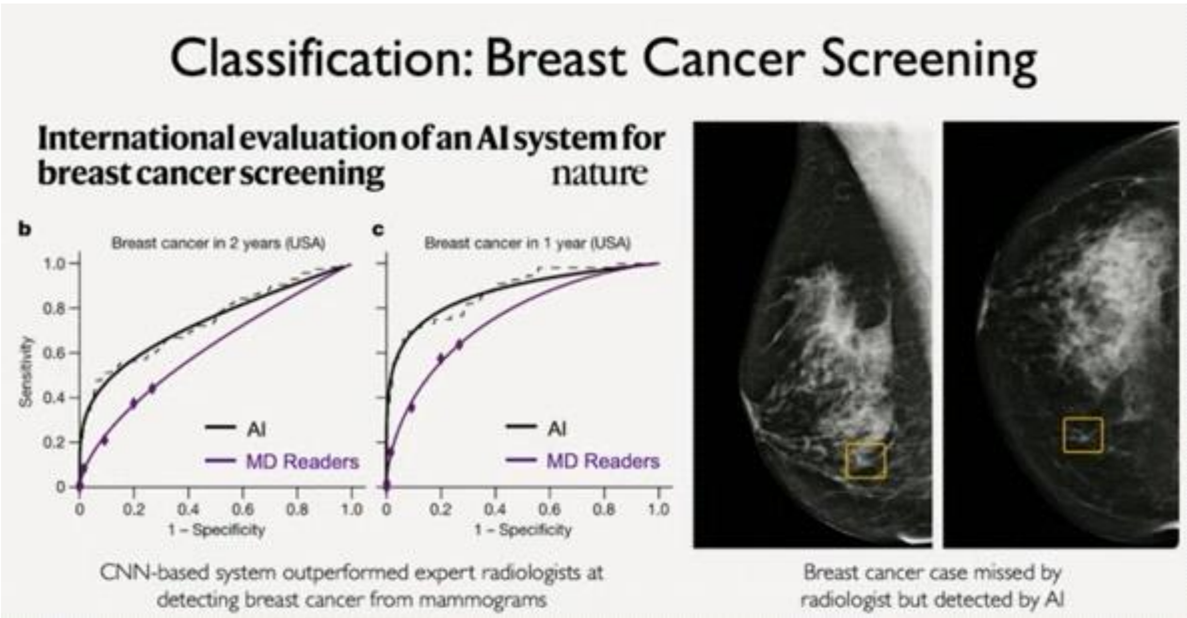


Possible convolutional applications

Now, the convolutional part of this network can be applied to many tasks beyond classification

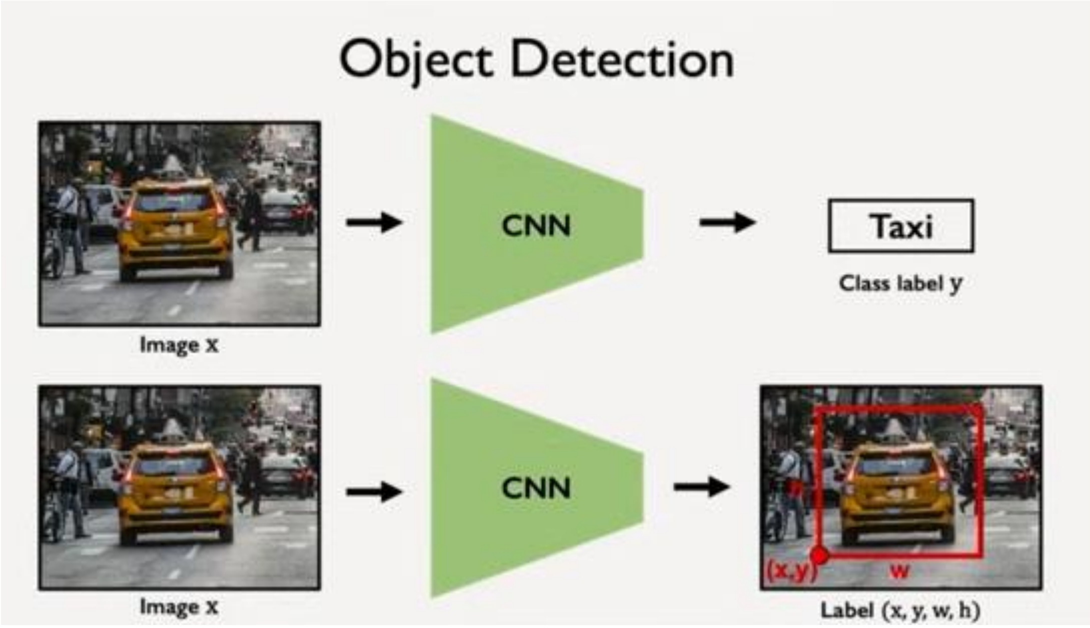


Example



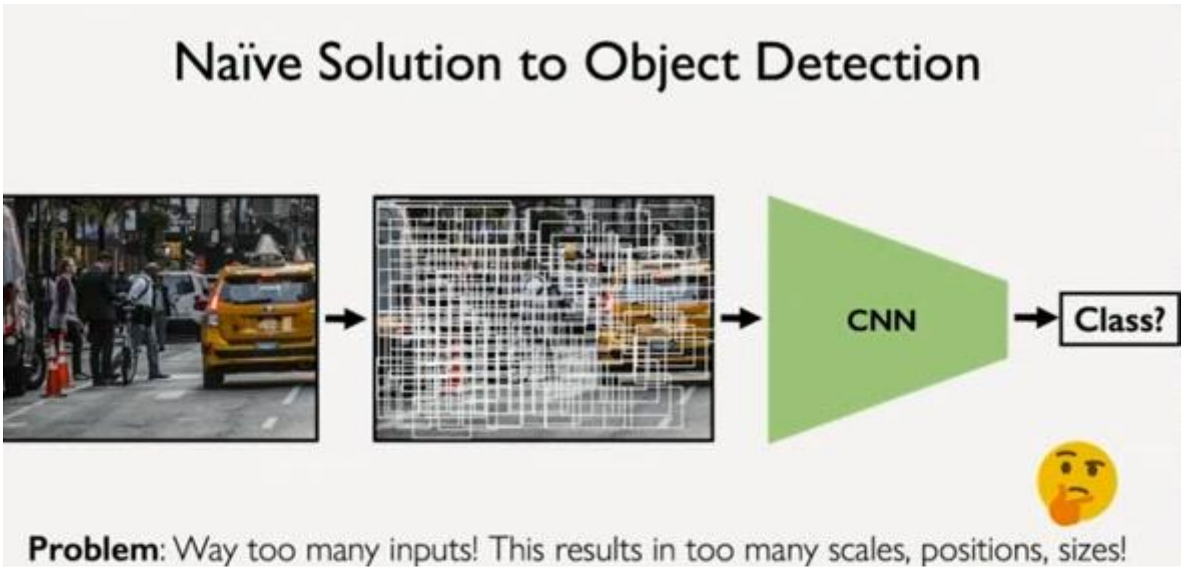
Example

Object detection predicting not only an object, but location, class and bounding boxes. The same for every other object.



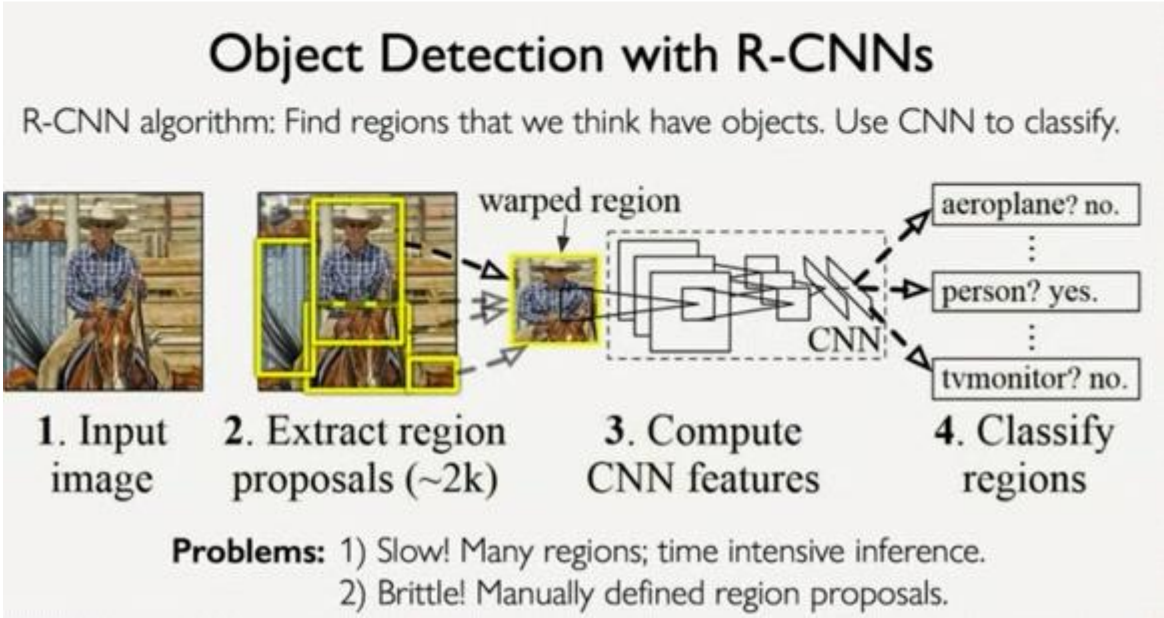
Object detection

Look at a naïve solution working on random boxes through CNNs and keep boxes with classes

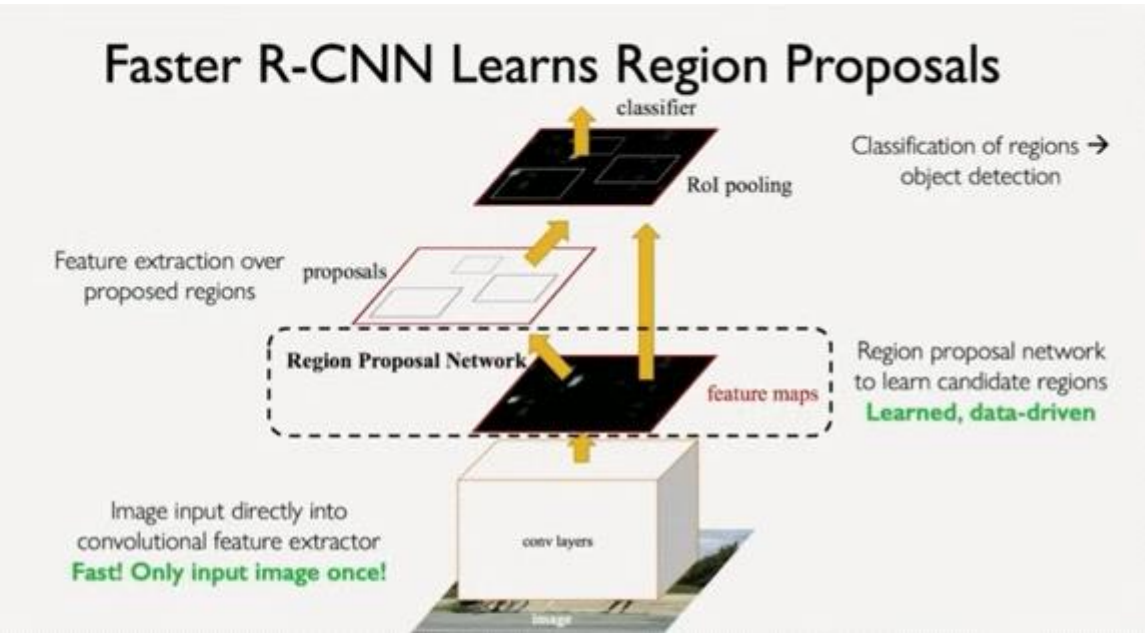


R-CNN

You can use heuristic suggesting regions, but it disconnets region proposal to the rest of the model and it is slow

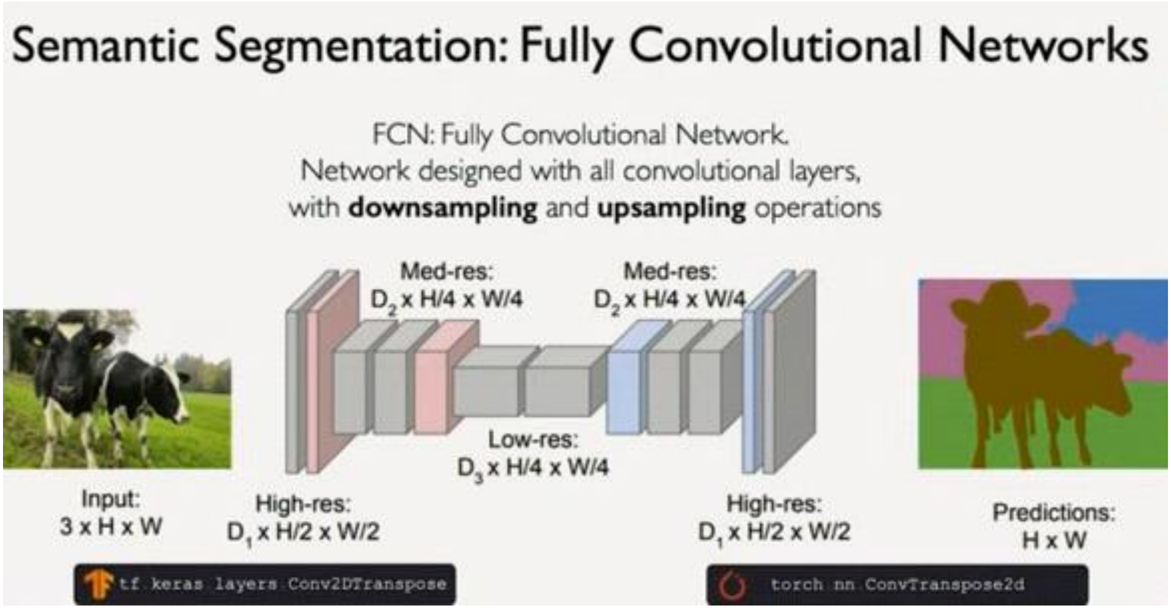


Faster R-CNN learns those regions within the same network as the classifier, so feature extraction happens on those regions

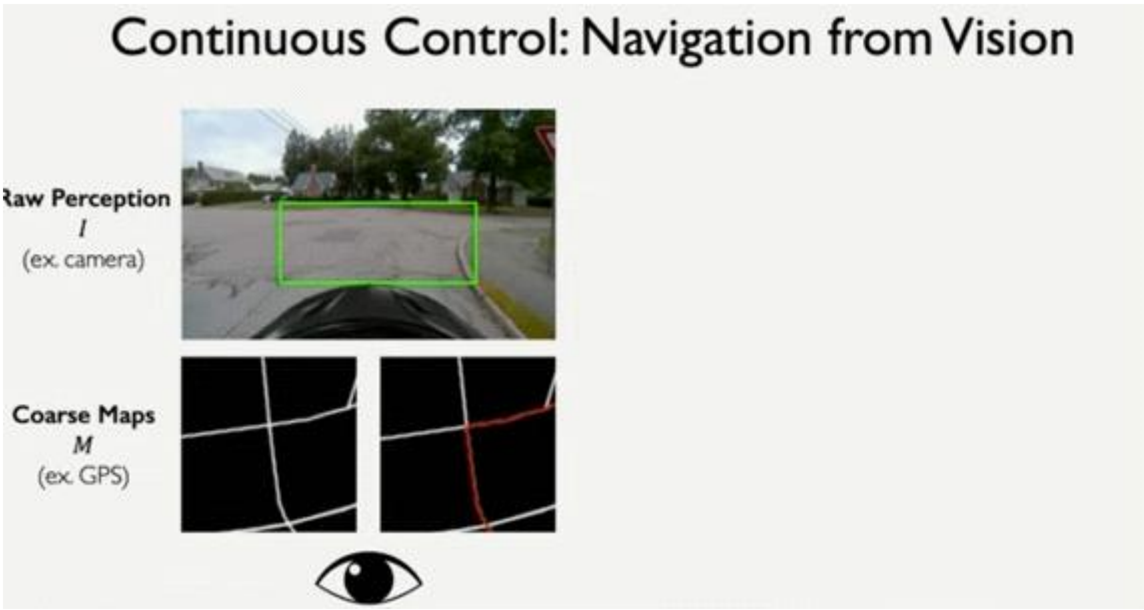


Segmentation

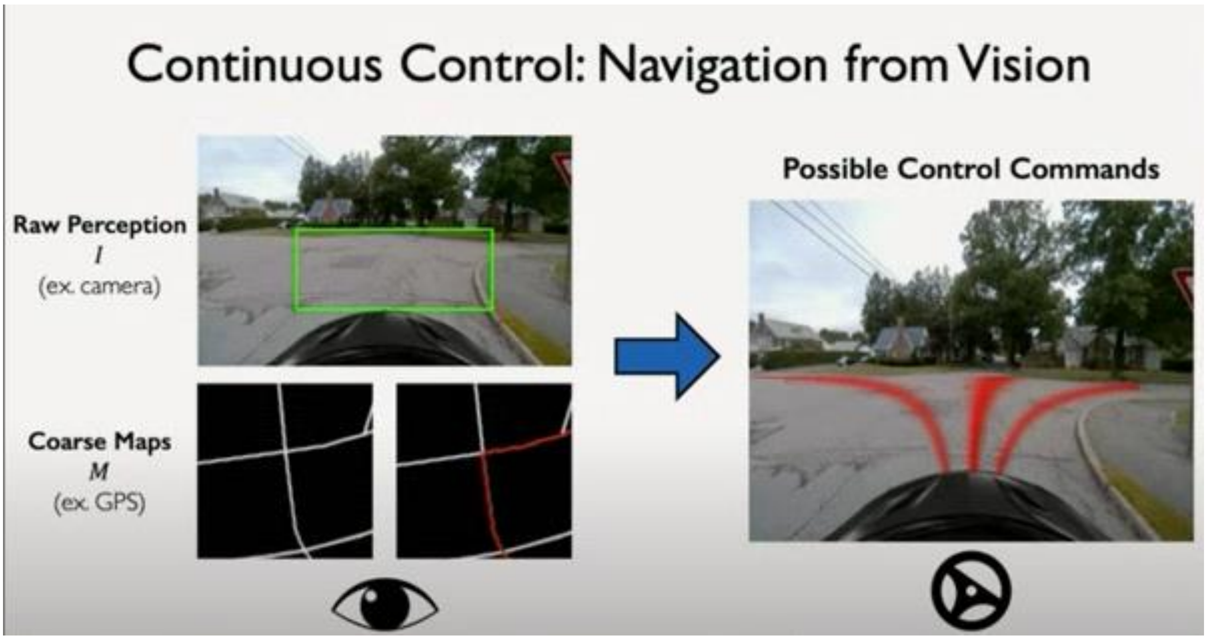
For every single pixel we want to learn another picture classifying every single pixel.
On the left we have convolution, pooling and non-linearity. On the right we have convolutions sampling up to the full image size again.

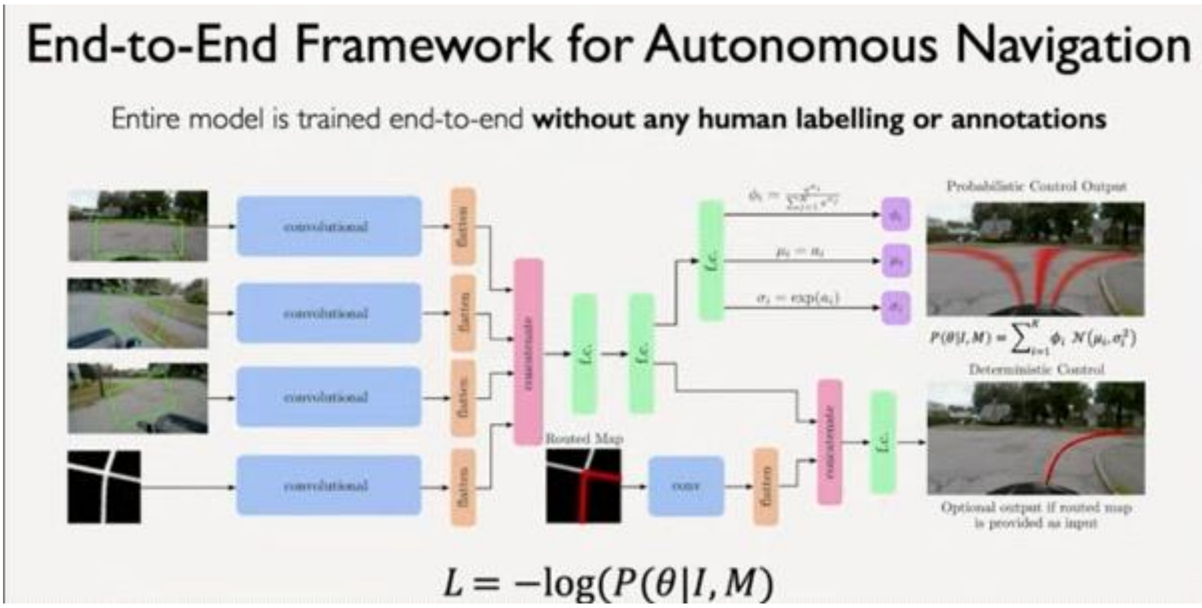


Automatic driving



We want to learn a continuous prob distribution of all possible control commands


Continuous Control: Navigation from Vision
Raw Perception
I
(ex. camera)
Coarse Maps
M
(ex. GPS)
Possible Control Commands


End-to-End Framework for Autonomous Navigation
Entire model is trained end-to-end **without any human labelling or annotations**
convolutional
flatten
concatenate
f.c.
Routed Map
conv
$\mu_i = a_i$
$\sigma_i = \exp(a_i)$
Probabilistic Control Output
$P(\theta|I,M) = \sum_{i=1}^{K} \phi_i \ N(\mu_i, \sigma_i^2)$
Deterministic Control
Optional output if routed map is provided as input
$L = -\log(P(\theta|I,M)$