


浙江大学爱丁堡大学联合学院

ZJU-UoE Institute

CNN for segmentation

Nicola Romanò - nicola.romano@ed.ac.uk

In the past few lectures...

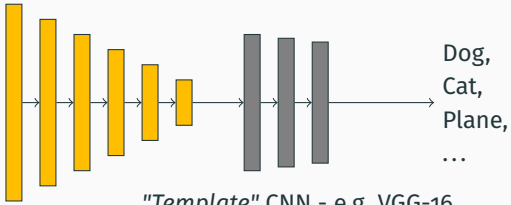
- We discussed ML approaches for image analysis.
- We introduced artificial neural networks (ANNs)
- We saw how convolutional neural networks (CNNs) can overcome the limitations of ANNs in image analysis.
- We discussed strategies for image classification by looking at classic examples (AlexNet, VGG, GoogLeNet, ...).

A note on transfer learning

- Training a CNN is expensive in terms of time and computer memory.
- Luckily, you can make use of pre-trained models to speed up the training process!
- This process is called **transfer learning**.
- For example, if you wanted to classify images, rather than start from scratch you could begin with VGG-16, or with GoogLeNet and modify them for your needs! (i.e. do not reinvent the wheel!)

A note on transfer learning

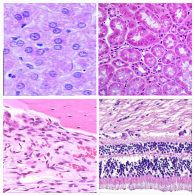
Generic images



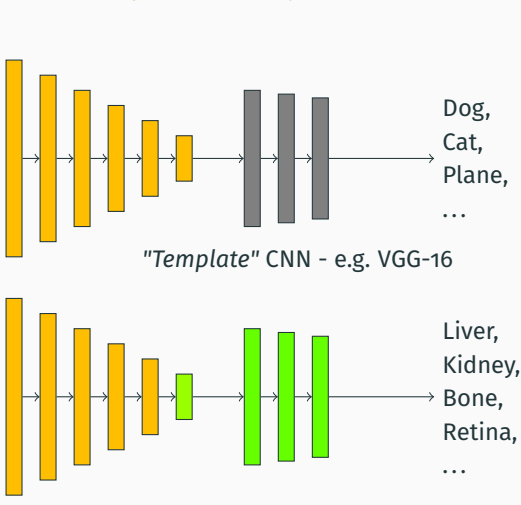
"Template" CNN - e.g. VGG-16

A note on transfer learning

Generic images

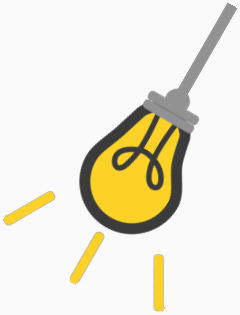


Task-specific images



Learning objectives

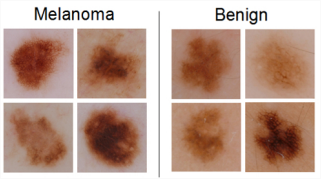
- Define the problems of segmentation, classification, localization and object detection.
- Describe CNN-based methods to solve these problems.
- Discuss advantages and disadvantages of these methods.



Introduction

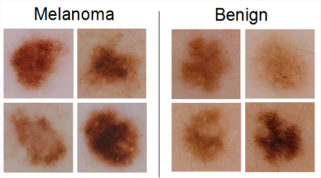
Computer vision problems

- **Image classification**: determine what class an image belongs to.

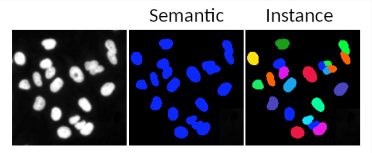


Computer vision problems

- **Image classification**: determine what class an image belongs to.

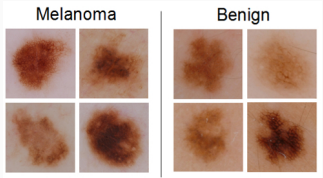


- **Segmentation**: dividing an image into groups (segments) of pixels with similar *meaning*.

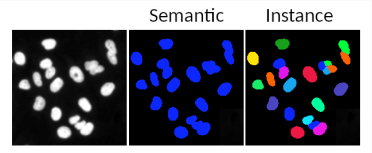


Computer vision problems

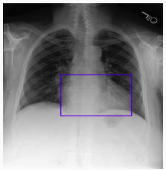
- **Image classification**: determine what class an image belongs to.



- **Segmentation**: dividing an image into groups (segments) of pixels with similar *meaning*.



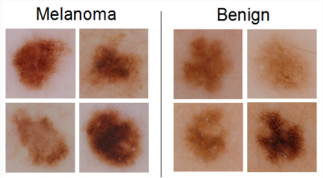
- **Localization**: determine the location (**bounding box**) of an object in an image.



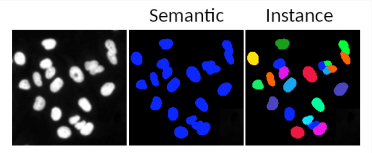
Nkechinyere et al. 2021

Computer vision problems

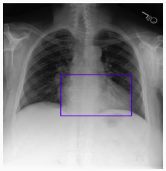
- **Image classification**: determine what class an image belongs to.



- **Segmentation**: dividing an image into groups (segments) of pixels with similar *meaning*.

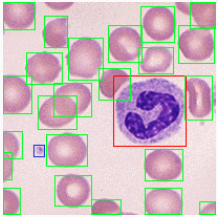


- **Localization**: determine the location (**bounding box**) of an object in an image.



Nkechinyere et al. 2021

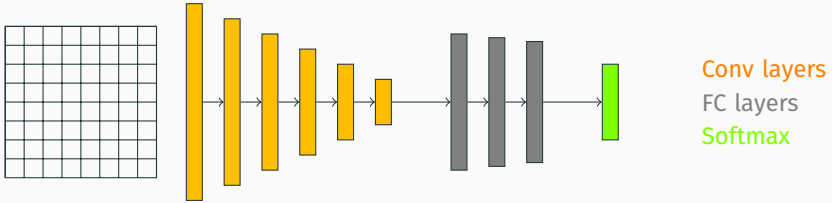
- **Detection**: Detecting the location of objects in an image, and classifying them.



Using CNN for segmentation

So far...

So far we've looked into CNNs for image classification

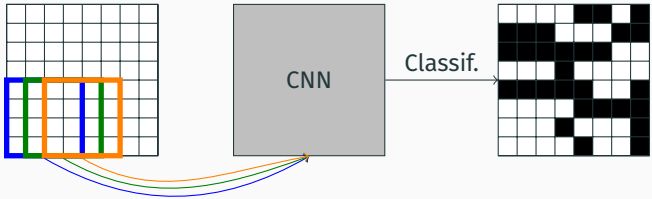


To segment an image, we need to use a different architecture, with an output that matches the input image.

How to do that?

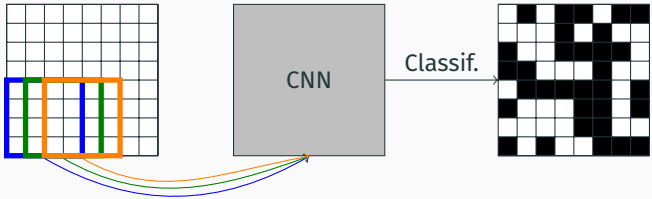
A naïve approach (don't do this!)

We could reduce the segmentation problem to a classification problem using a sliding window going over each pixel and then through a CNN.



A naïve approach (don't do this!)

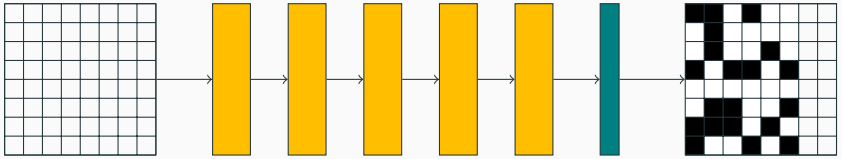
We could reduce the segmentation problem to a classification problem using a sliding window going over each pixel and then through a CNN.



- Slow and computationally intensive
- Inefficient: we reuse information from the same pixel multiple times

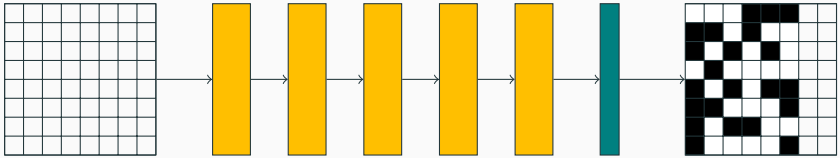
A better convolutional-only approach

We can use a series of same convolutions to keep image size, then use 1×1 convolutions to reduce the layer depth to the desired number of classes c.



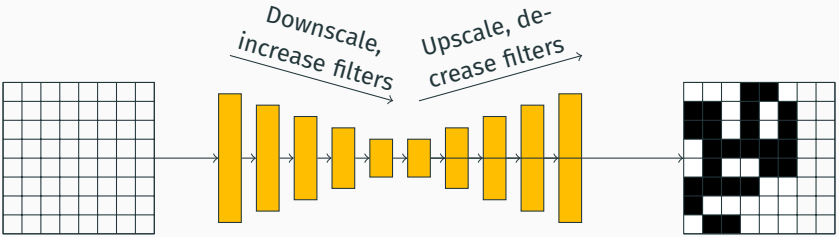
A better convolutional-only approach

We can use a series of same convolutions to keep image size, then use 1×1 convolutions to reduce the layer depth to the desired number of classes c.

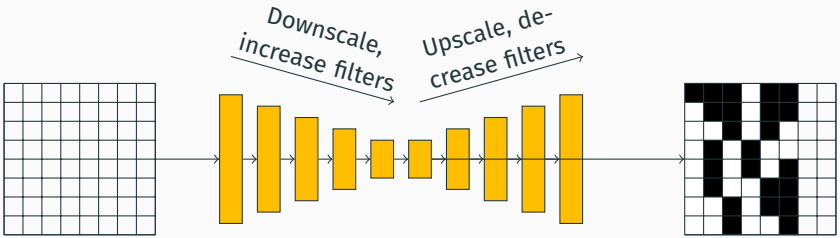


- Faster and more efficient
- Still a lot of calculations (same convolutions, no pooling), so problematic for big images

Downscale, then upscale!



Downscale, then upscale!



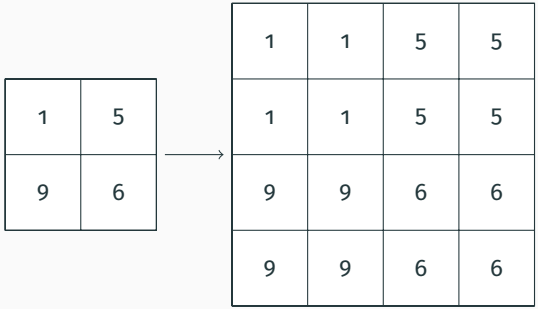
Advantages:

- Convolutional only system
- Most of the processing is done at a small spatial scale, so it is faster
- The downscaling path helps determining the "what" while the upscale path helps determining the "where".

Upscaling - non-learnable

Simple ideas for upscaling include **nearest neighbor interpolation**, **bilinear interpolation**, or "**bed of nails**" approaches.

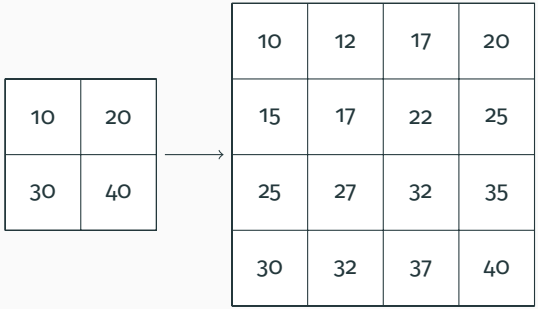
Nearest neighbour interpolation



Upscaling - non-learnable

Simple ideas for upscaling include **nearest neighbor interpolation**, **bilinear interpolation**, or "**bed of nails**" approaches.

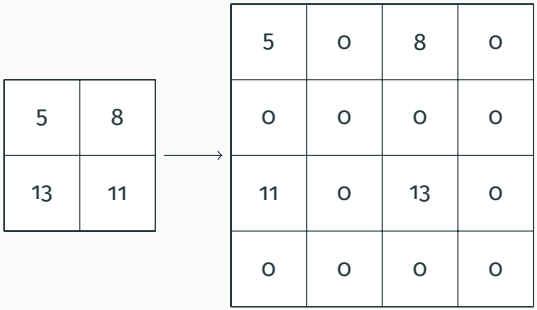
Bilinear interpolation



Upscaling - non-learnable

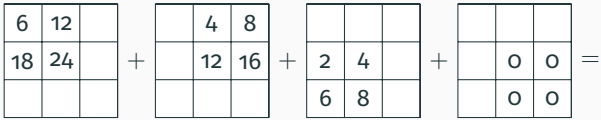
Simple ideas for upscaling include **nearest neighbor interpolation**, **bilinear interpolation**, or "**bed of nails**" approaches.

Bed of nails



Upscaling - learnable

A learnable approach to upscaling is **transpose convolutions**. The value of each pixel is used as a scaler for a learnable filter.



Input

6	4
2	0

Learnable kernel

1	2
3	4

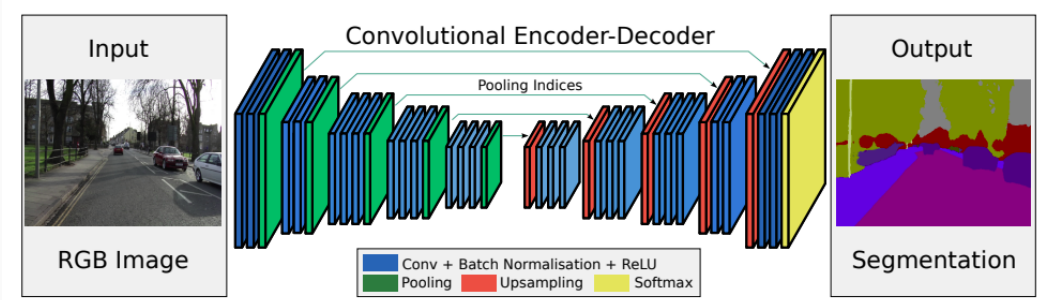
Output

6	16	8
20	40	16
6	8	0

Advantage: learnable approach

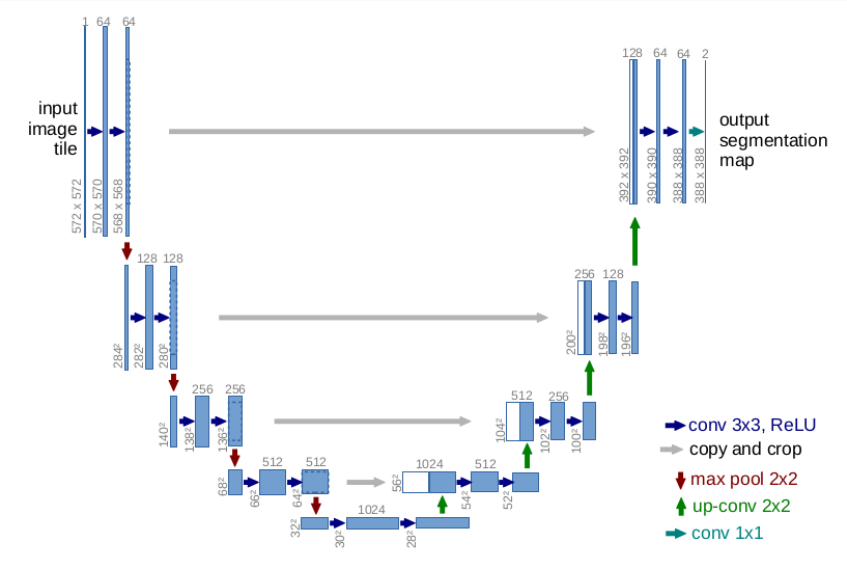
Downside: can lead to "checkerboard artefacts" if the stride/size is not chosen correctly.

Example - SegNet



Badrinarayanan, A., et al. **SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation**.

U-Net - segmentation for biomedical images



Ronneberger et al., 2015 - U-Net: Convolutional Networks for Biomedical Image Segmentation

Localization and detection

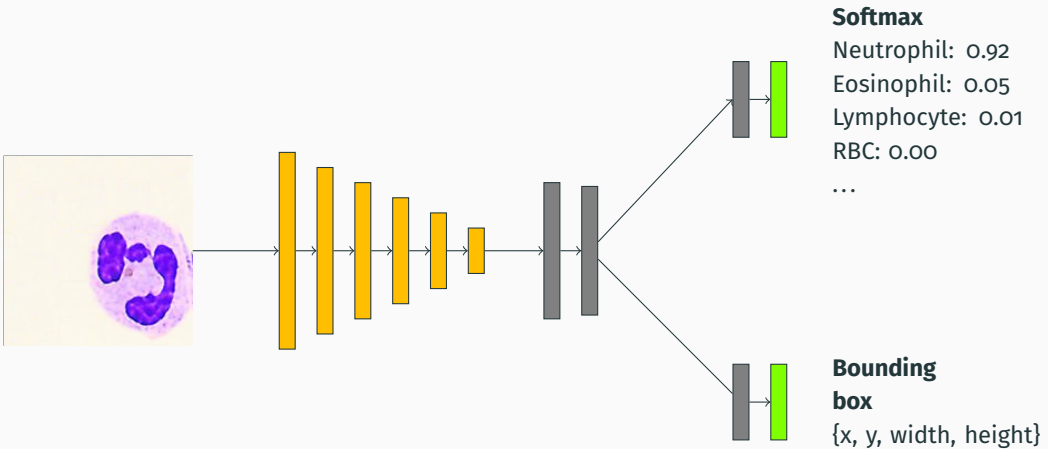
Localization

Localization problem: find the position of an object of interest in the image. We know that there is exactly one.

For example, I want to find the position of a cell in an image, and define what type of cell that is.

How to do that?

CNN with multiple outputs



This network will use two losses, one for the classification and one for the bounding box. We will have an hyperparameter to weight the two losses at training time.

Which losses do we use for these problems?

Classification

- Most commonly function is the **cross-entropy** loss
- Often called **softmax loss**. This is actually a softmax activation followed by a cross-entropy loss!
- For binary problems a variant is the **binary cross-entropy**

Cross entropy is defined as

$$L(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i)$$

where $\hat{y}_i$ is the predicted probability of the i-th class and y is the ground truth.

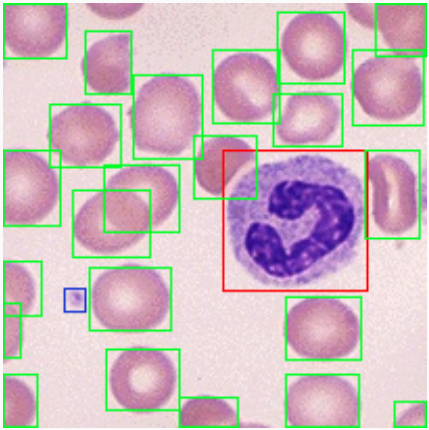
Which losses do we use for these problems?

Bounding box We can treat this as a regression problem and use a **mean squared error loss** (also called **L_2 loss**).

$$L(y, \hat{y}) = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

The detection problem

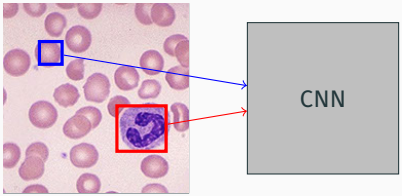
What if we have multiple objects?



Problematic, as we don't know how many, if any, objects we are looking for!

Cropping the image

A simple idea is to generate several crops of the image, and classify them separately.



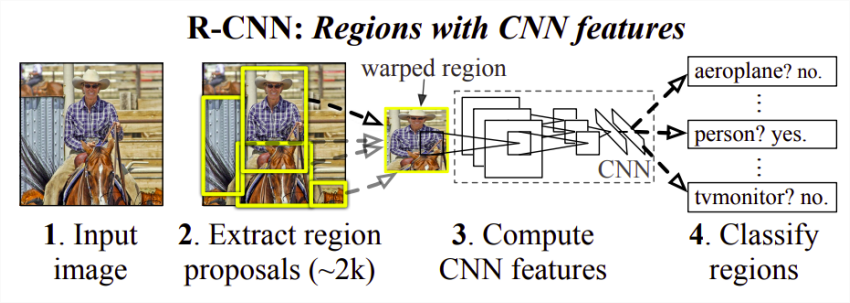
Neutrophil: 0.1
Platelet: 0.05
Lymphocyte: 0.01
RBC: 0.84
Background:0
+
{0.2, 0.2, 0.1, 0.1}

Neutrophil: 0.96
Platelet: 0.02
Lymphocyte: 0.01
RBC: 0.01
Background:0
+
{0.6, 0.6, 0.3, 0.25}

Problem: how do I choose the crops?

R-CNN

- R-CNN: region proposals + CNN
- Generates 2000 regions
- The original paper uses the "selective search" algorithm, but R-CNN works with any other region searching algorithm
- Crops for the selected regions are passed through a CNN

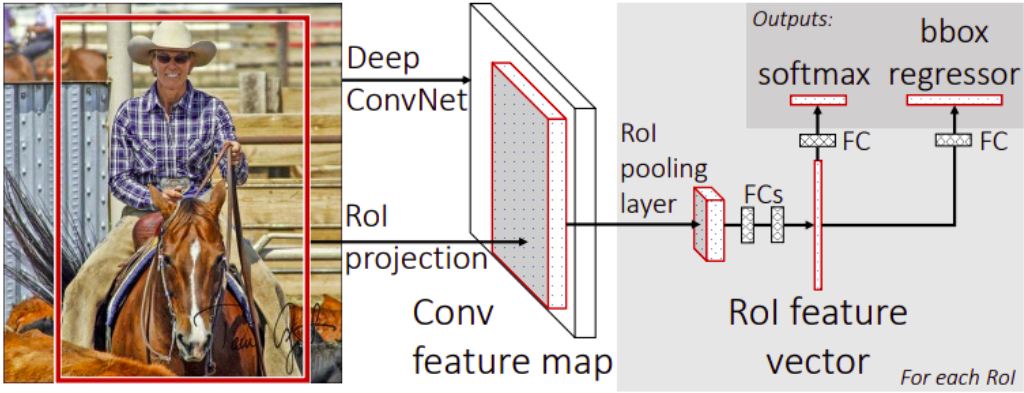


Girshick et al. 2014

Region proposal is fast (few seconds), but passing 2000 regions to a CNN is still time consuming.

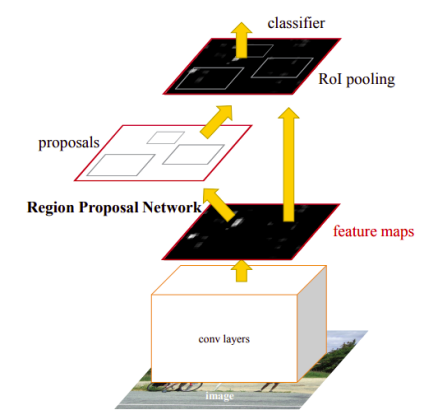
Fast R-CNN

- We propose regions as in R-CNN
- We pass the whole image through a CNN only once
- We can then crop the feature map using the proposed regions saving a lot of computation!
- 10x faster at training time, 150x faster at run time!



Faster R-CNN

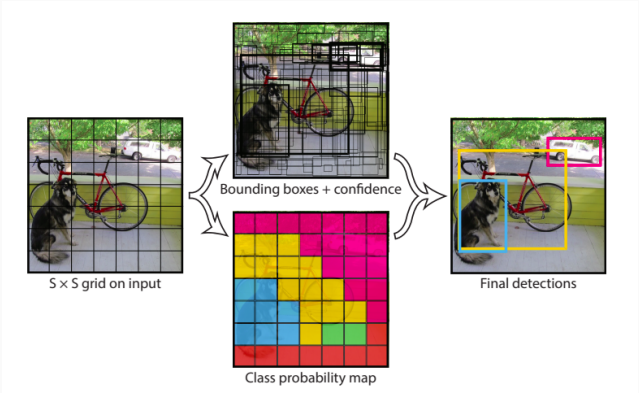
- We pass the whole image through a CNN
- We learn regions from the feature map using a fully convolutional network
- We then crop the feature map as in Fast R-CNN
- 200+ times faster than R-CNN



Ren et al., 2016

YOLO

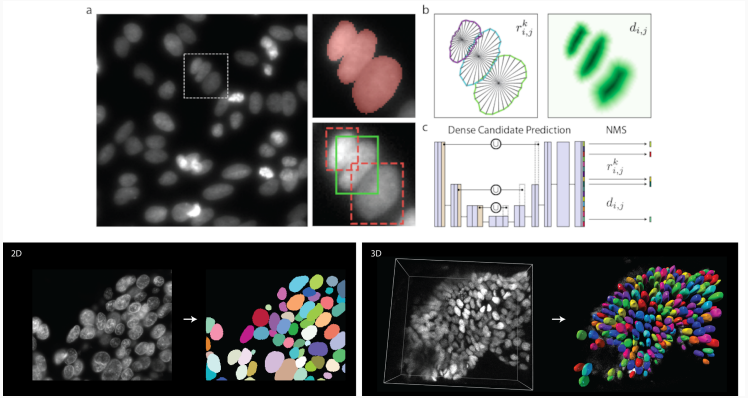
- YOLO splits the image into a grid of cells
- For each cell it generate a series of bounding boxes prediction and a class probability
- It's a one-shot method, so very fast!
- Strong spatial contstraints. Each cell can only predict a single object.



Redmon et al., 2016

A final example: Stardist for cell segmentation

The UNet architecture has been heavily used and modified. For example, the Stardist network is a modification of the UNet architecture for 2D and 3D segmentation of cells.



Schmidt et al. - Cell Detection with Star-convex Polygons. Weigert et al. - Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy.