DEEP LEARNING FOR OBJECT DETECTION IN COMPUTER VISION

From CNNs to Mask R-CNNs: A brief history

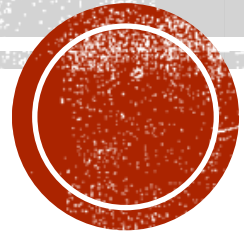
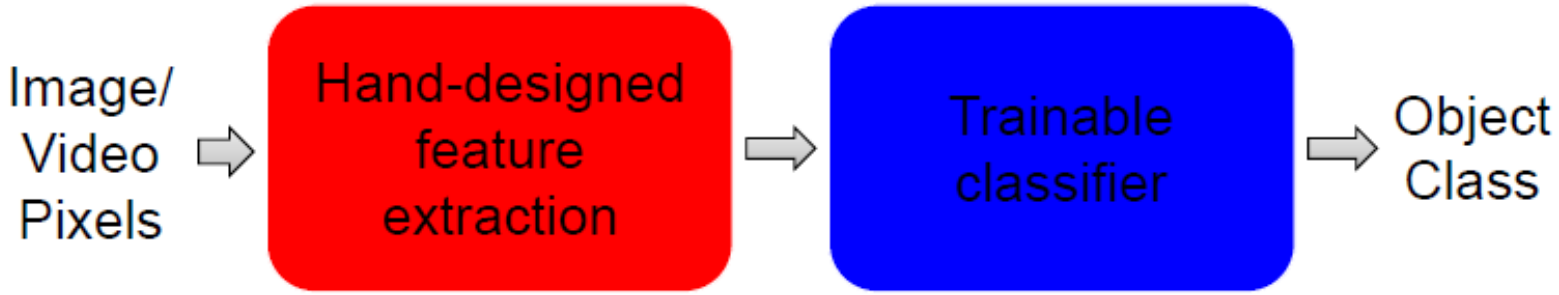


IMAGE CLASSIFICATION AND OBJECT DETECTION BEFORE DEEP LEARNING

Traditional Recognition Approach



- Features are not learned
- Trainable classifier is often generic (e.g. SVM)

THE IDEA BEHIND CNN

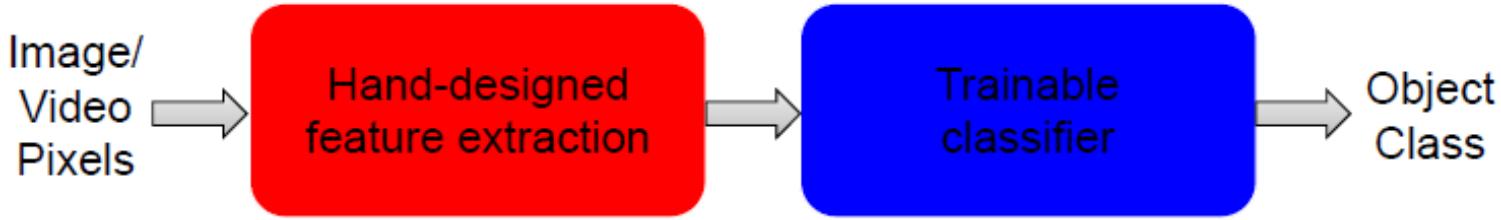
What about learning the features?

- Learn a *feature hierarchy* all the way from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly

DEEP VS "SHALLOW" ARCHITECTURES

"Shallow" vs. "deep" architectures

Traditional recognition: "Shallow" architecture

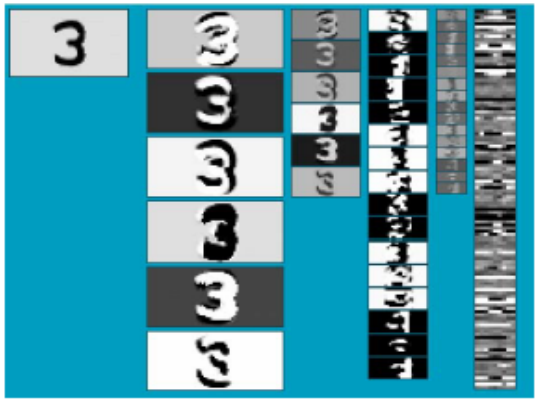


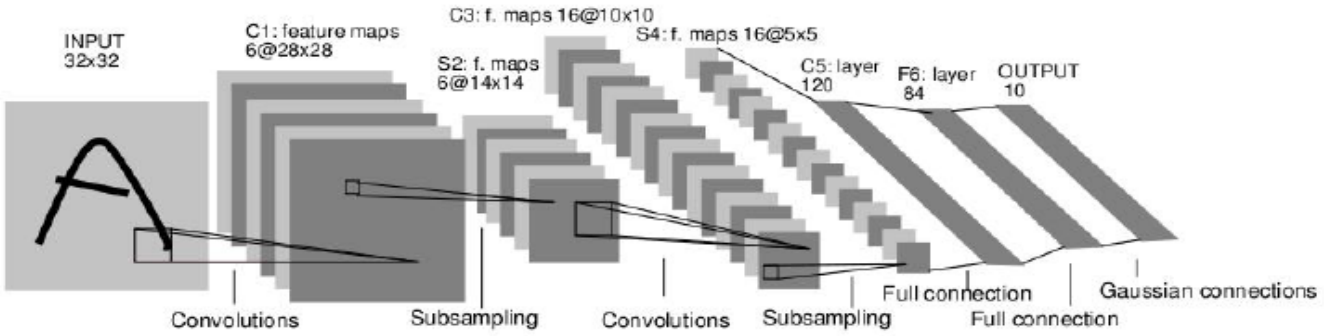
Deep learning: "Deep" architecture

CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional Neural Networks (CNN, Convnet)

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant features
- Classification layer at the end





Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

CNN: FIRST IMPRESSIVE SUCCESSES

Convnet Successes

- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011])
  - Arabic & Chinese [Ciresan et al. 2012]
- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 2011]
- But until recently, less good at more complex datasets
  - Caltech-101/256 (few training examples)

CNN: THE IMAGENET CHALLENGE 2012

ImageNet Challenge 2012

IMAGENET

[Deng et al. CVPR 2009]

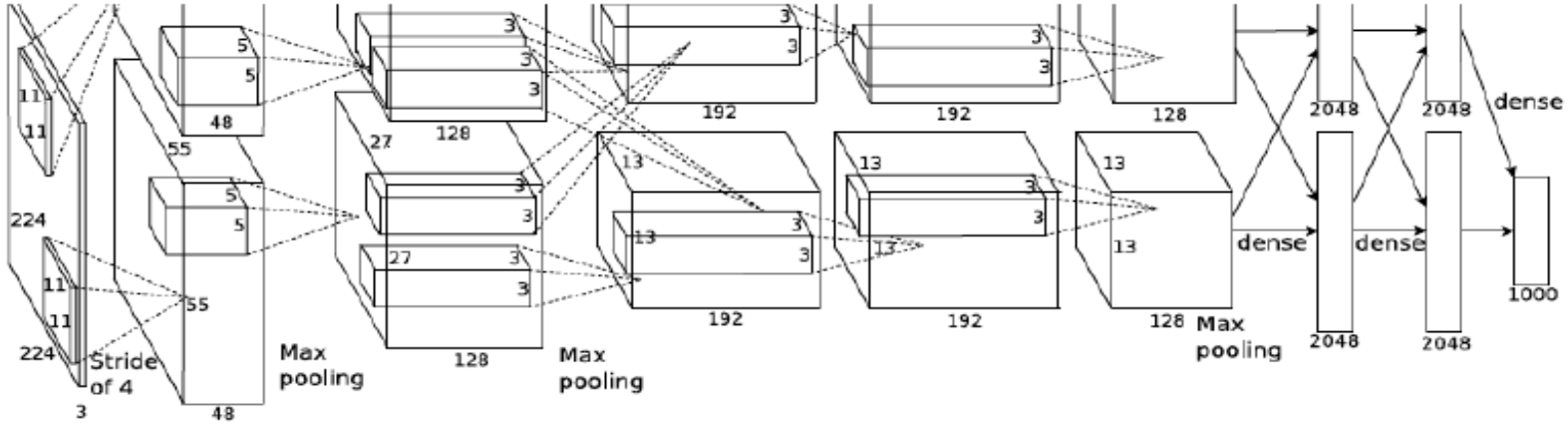
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk
- Challenge: 1.2 million training images, 1000 classes

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

CNN: THE IMAGENET CHALLENGE 2012

ImageNet Challenge 2012

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Better regularization for training (DropOut)



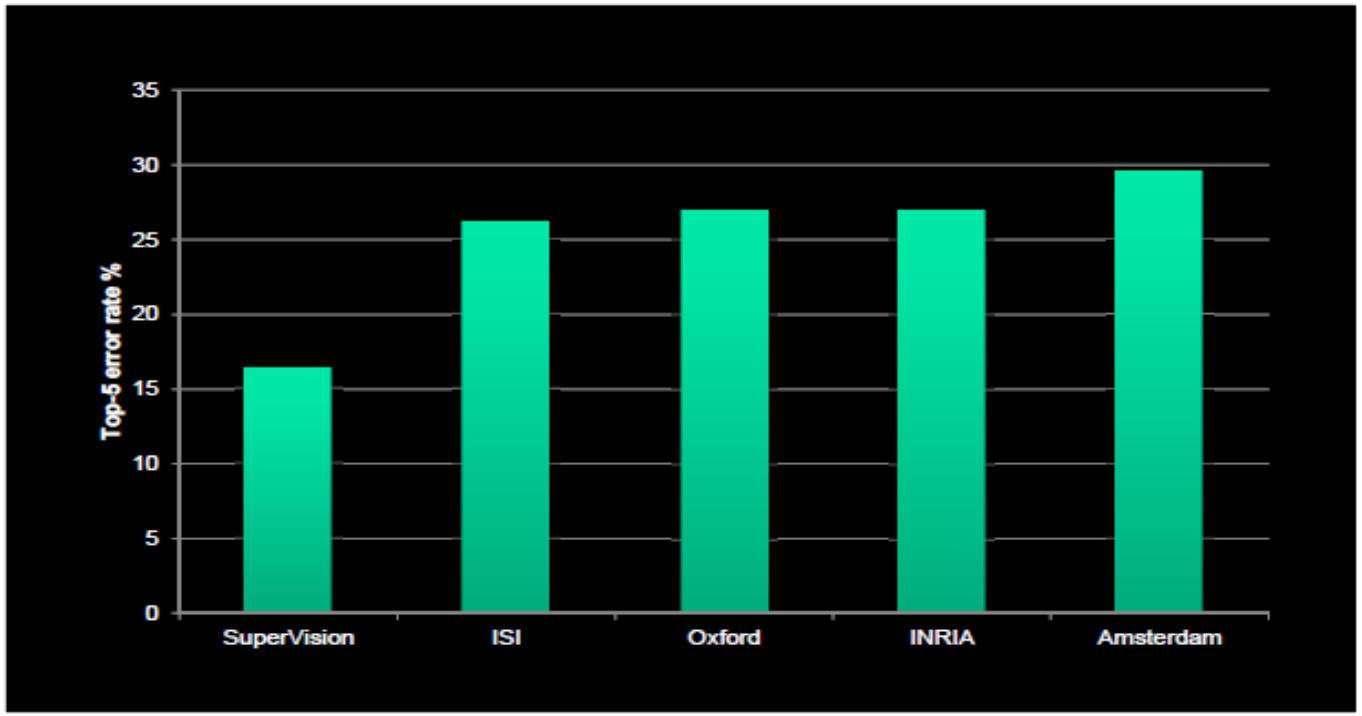
A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

CNN: THE IMAGENET CHALLENGE 2012

ImageNet Challenge 2012

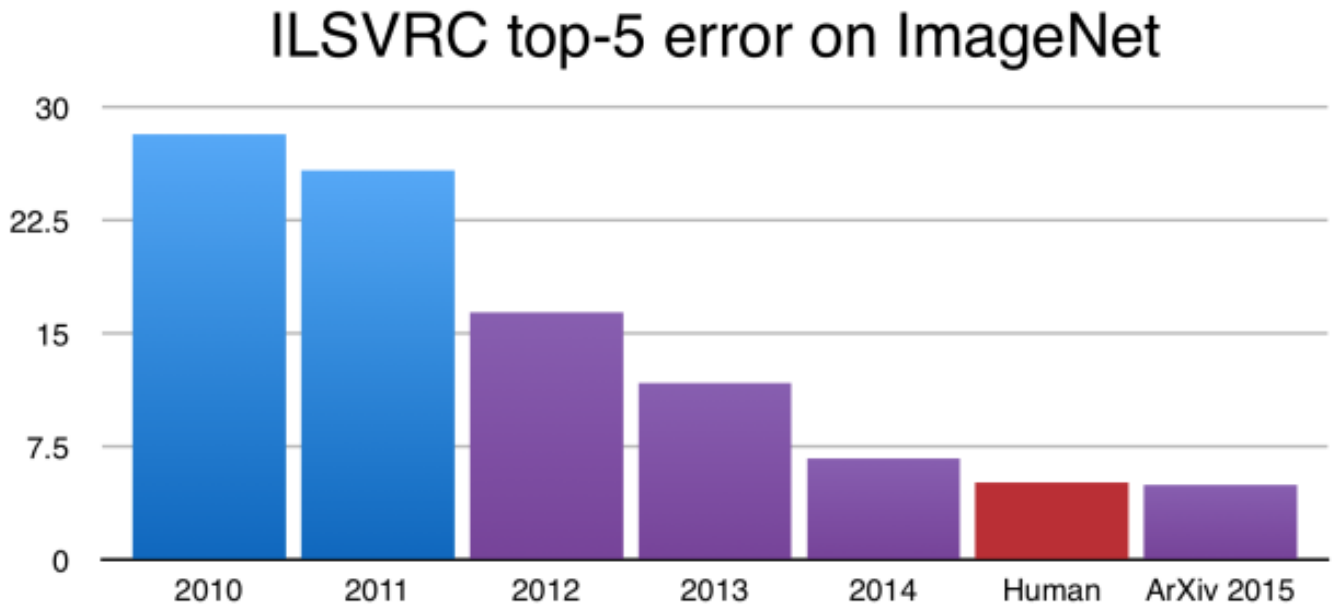
Krizhevsky et al. -- **16.4% error** (top-5)

Next best (non-convnet) – **26.2% error**



SO, CNN: THE GOLD STANDARD FOR IMAGE CLASSIFICATION

- Convolutional Neural Networks(CNNs) have become the gold standard for image classification.
- In fact, since then, CNNs have improved to the point where they now outperform humans on the ImageNet challenge!

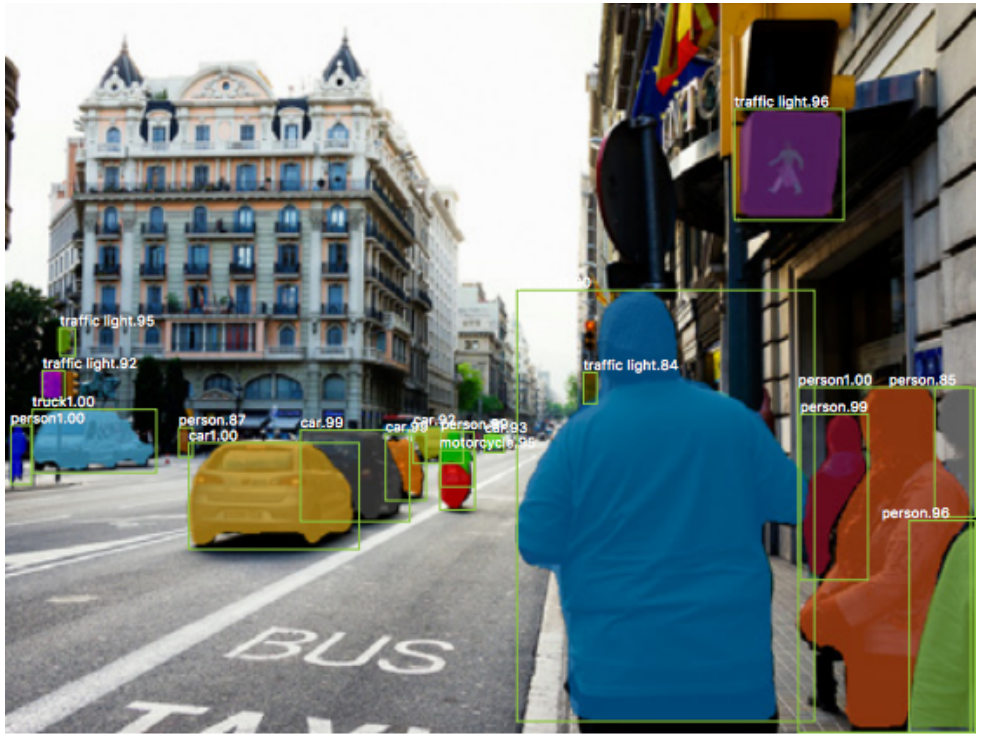


HOWEVER, IMAGE CLASSIFICATION ≠ OBJECT DETECTION

- While these results are impressive, image classification is far simpler than the complexity and diversity of true human visual understanding.
- In classification, there's generally an image with a single object as the focus and the task is to say what that image is. But when we look at the world around us, we carry out far more complex tasks.

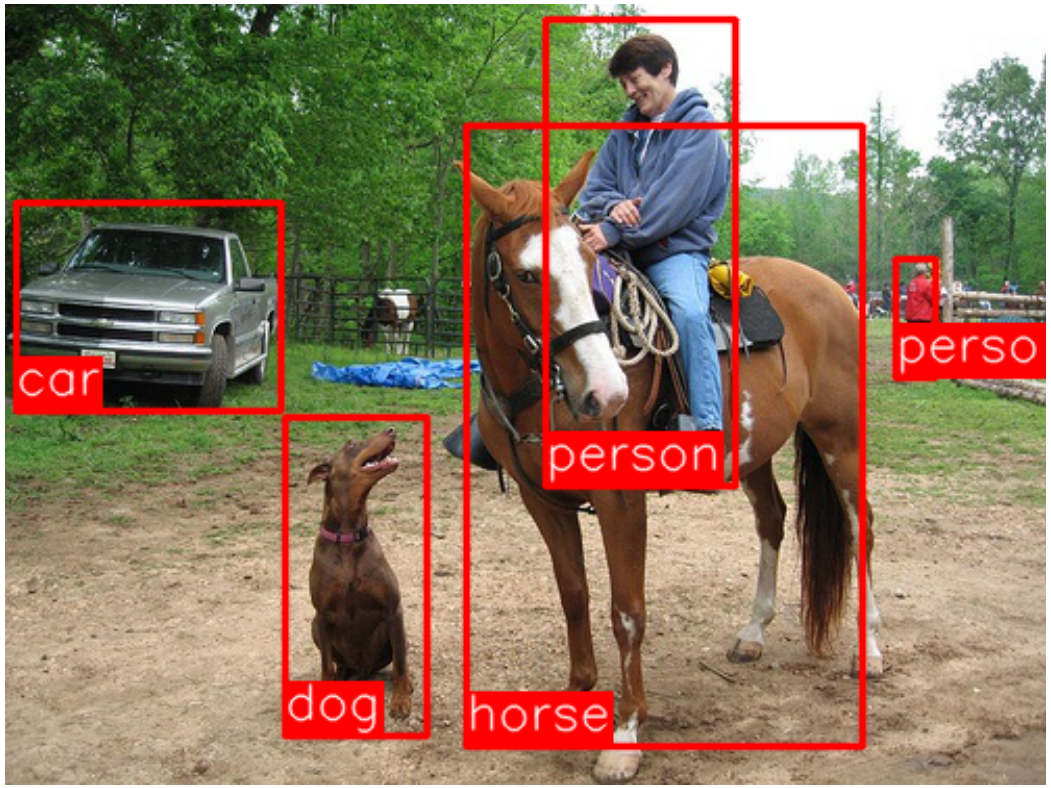
SO, CNN FOR OBJECT DETECTION?

- We see complicated sights with multiple overlapping objects, and different backgrounds and we not only classify these different objects but also identify their boundaries, differences, and relations to one another!
- Can CNNs help us with such complex tasks? Namely, given a more complicated image, can we use CNNs to identify the different objects in the image, and their boundaries?



R-CNN: AN EARLY APPLICATION OF CNN TO OBJECT DETECTION (2014)

- Inspired by the research of Hinton's lab at the University of Toronto, a small team at UC Berkeley, led by Professor Jitendra Malik, asked themselves:
- *To what extent do [Krizhevsky et. Al]'s results on CNN-based image classification generalize to object detection?*
- As their 2014 paper showed, the answer is *dramatically*
- Their approach showed dramatically higher object detection performance on PASCAL VOC as compared to systems based on simpler HOG-like features.



R-CNN: HOW IT WORKS

- The goal of R-CNN is to take in an image, and correctly identify where the main objects (via a bounding box) in the image.
- **Inputs**: Image
- **Outputs**: Bounding boxes + labels for each object in the image.
- But how do we find out where these bounding boxes are?
- R-CNN **proposes a bunch of boxes in the image and see if any of them actually correspond to an object**.
- R-CNN creates these bounding boxes, or region proposals, using a process called **Selective Search**
- At a high level, Selective Search looks at the image through windows of different sizes, and for each size tries to group together adjacent pixels by texture, color, or intensity to identify objects.

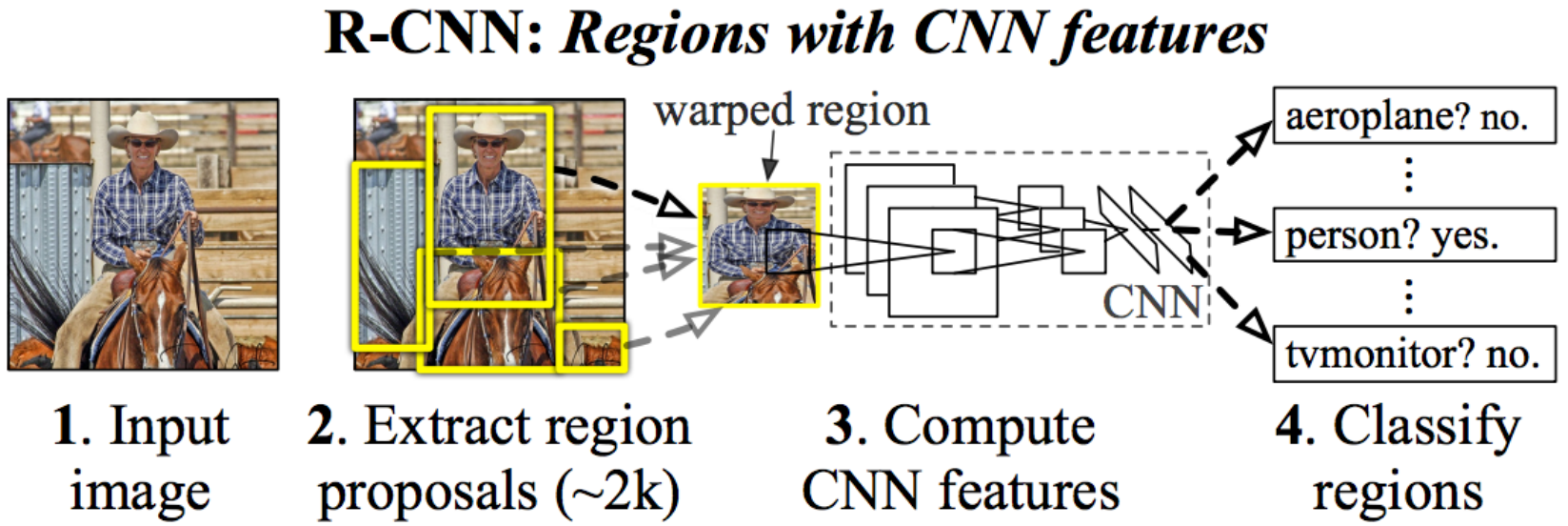
R-CNN: HOW IT WORKS

- Selective Search looks through windows of multiple scales and looks for adjacent pixels that share textures, colors, or intensities.

Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.

R-CNN: HOW IT WORKS

- Once the proposals are created, R-CNN warps the region to a standard square size and passes it through to a modified version of AlexNet (the winning submission to ImageNet 2012 that inspired R-CNN).
- On the final layer of the CNN, R-CNN adds a Support Vector Machine (SVM) that simply classifies whether this is an object, and if so what object. This is step 4.



R-CNN: HOW IT WORKS

- Now, having found the object in the box, can we tighten the box to fit the true dimensions of the object?
- We can, and this is the final step of R-CNN. R-CNN runs a simple linear regression on the region proposal to generate tighter bounding box coordinates to get our final result. Here are the inputs and outputs of this regression model:
- **Inputs**: sub-regions of the image corresponding to objects.
- **Outputs**: New bounding box coordinates for the object in the sub-region.
- So, to summarize, R-CNN is just the following steps:
  - Generate a set of proposals for bounding boxes.
  - Run the images in the bounding boxes through a pre-trained AlexNet and finally an SVM to see what object the image in the box is.
  - Run the box through a linear regression model to output tighter coordinates for the box once the object has been classified.

FAST R-CNN: SPEEDING UP AND SIMPLIFYING R-CNN (2015)

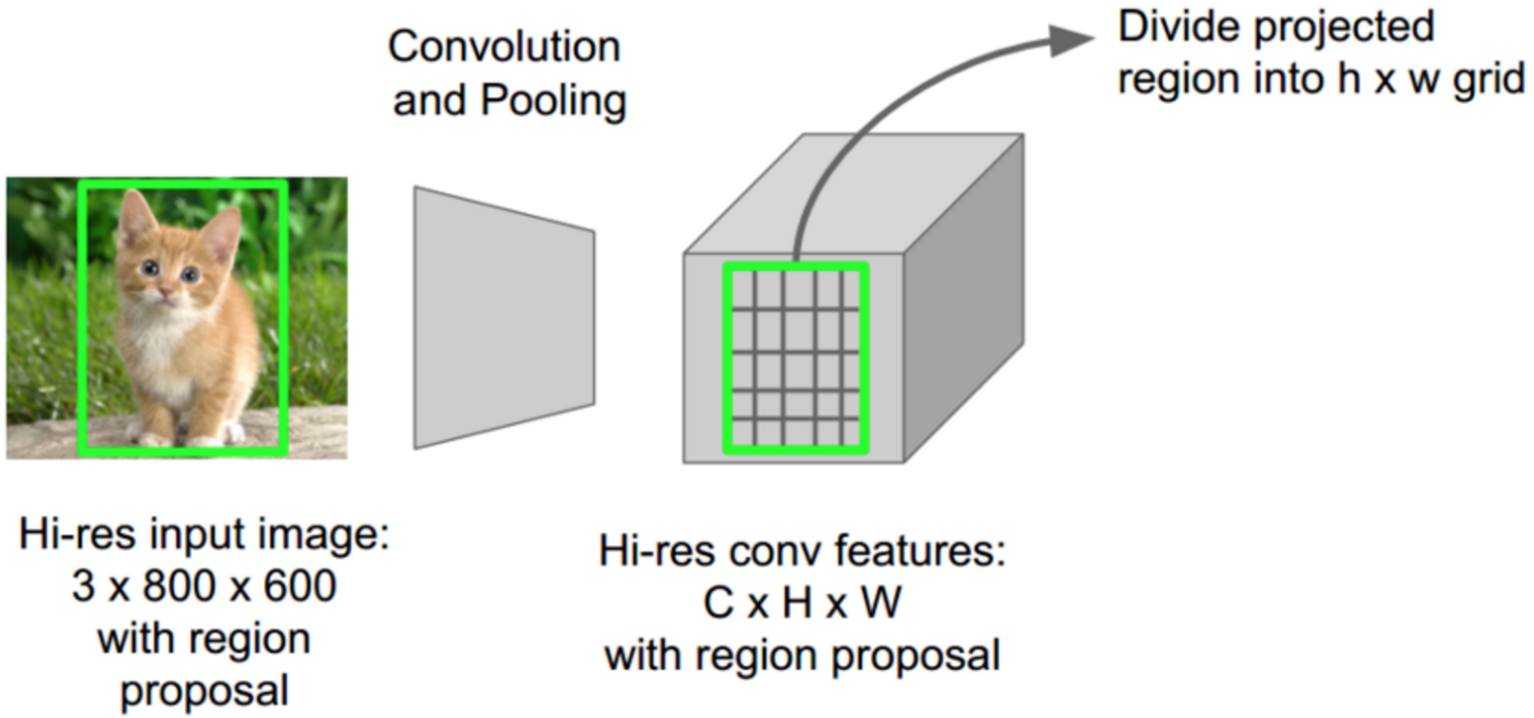
- R-CNN works really well, but is really quite slow for a few simple reasons:
- It requires a forward pass of the CNN (AlexNet) for every single region proposal for every single image (that's around 2000 forward passes per image!).
- It has to train three different models separately - the CNN to generate image features, the classifier that predicts the class, and the regression model to tighten the bounding boxes. This makes the pipeline extremely hard to train.
- Solution to the above problems: Fast R-CNN by Ross Girshick (2015)

FAST R-CNN: ROI POOLING

- For the forward pass of the CNN, Girshick realized that for each image, a lot of proposed regions for the image invariably overlapped causing us to run the same CNN computation again and again (~2000 times!).
- His insight was simple: **Why not run the CNN just once per image and then find a way to share that computation across the ~2000 proposals?**
- This is exactly what Fast R-CNN does using a technique known as RoIPool (Region of Interest Pooling).
- At its core, RoIPool shares the forward pass of a CNN for an image across its subregions. In the following image, the CNN features for each region are obtained by selecting a corresponding region from the CNN's feature map. Then, the features in each region are pooled (usually using max pooling). So all it takes us is one pass of the original image as opposed to ~2000!

FAST R-CNN: ROI POOLING

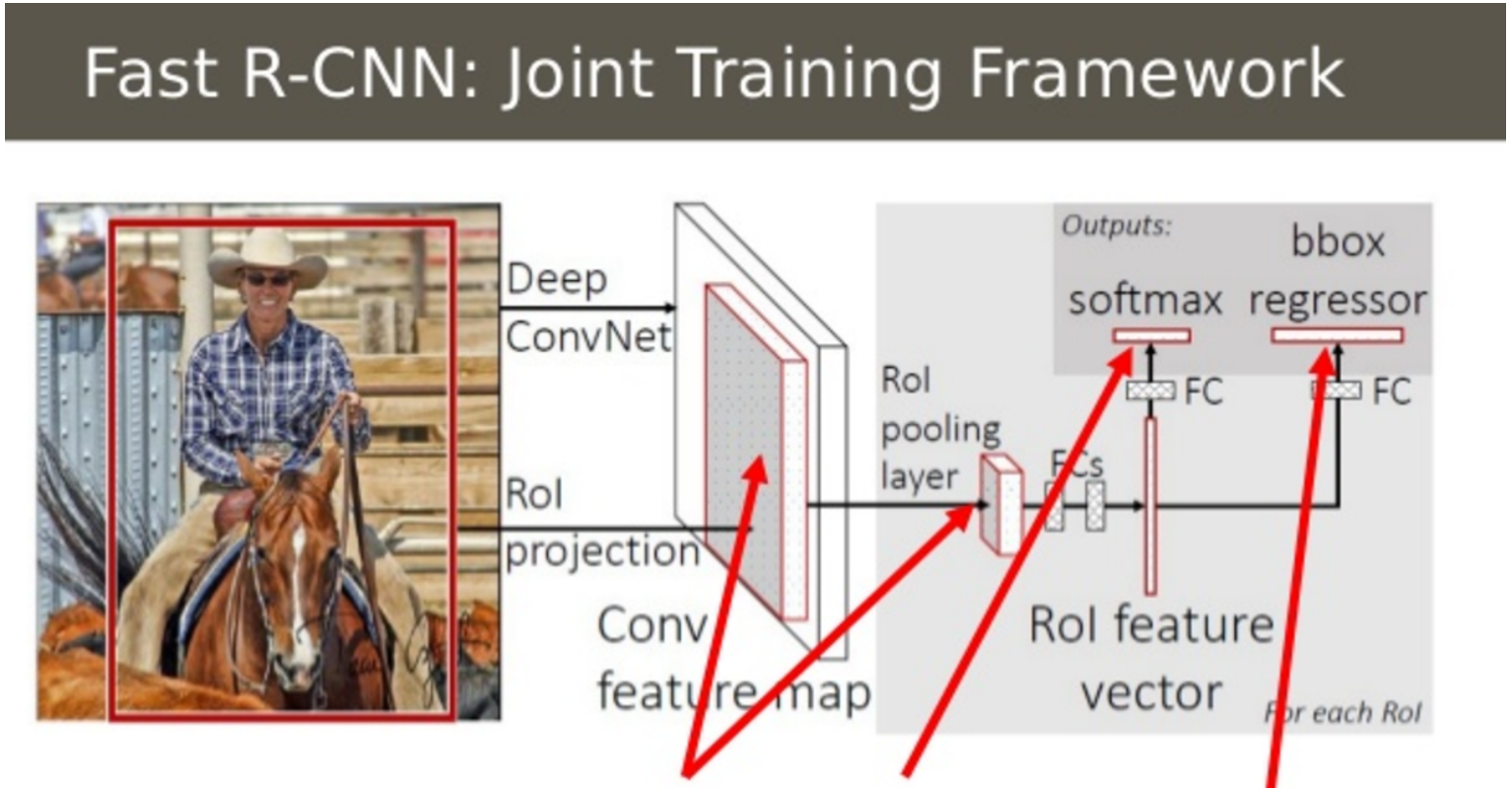
- In RoIPool, a full forward pass of the image is created and the conv features for each region of interest are extracted from the resulting forward pass.



FAST R-CNN: COMBINING ALL MODELS IN ONE NETWORK

- The second insight of Fast R-CNN is to jointly train the CNN, classifier, and bounding box regressor in a single model.
- Where earlier we had different models to extract image features (CNN), classify (SVM), and tighten bounding boxes (regressor), **Fast R-CNN instead used a single network to compute all three.**
- Fast R-CNN replaced the SVM classifier with a softmax layer on top of the CNN to output a classification. It also added a linear regression layer parallel to the softmax layer to output bounding box coordinates. In this way, all the outputs needed came from one single network!
- Here are the inputs and outputs to this overall model:
  - **Inputs**: Images with region proposals.
  - **Outputs**: Object classifications of each region along with tighter bounding boxes.

FAST R-CNN: COMBINING ALL MODELS IN ONE NETWORK



FASTER R-CNN: SPEEDING UP REGION PROPOSAL

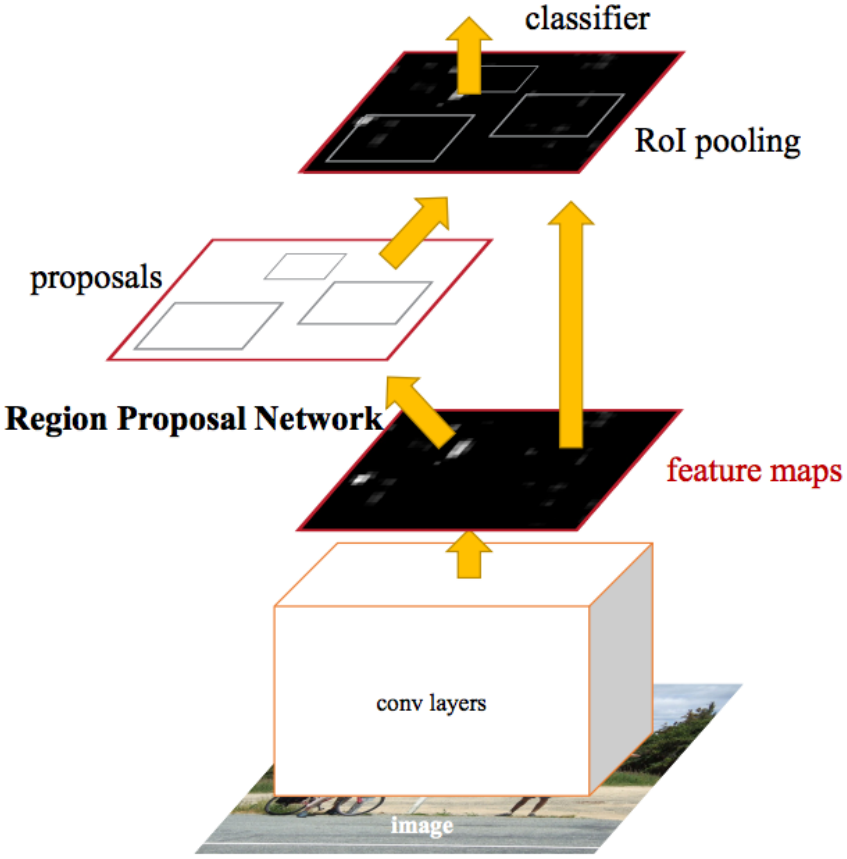
- Even with all these advancements, there was still one remaining bottleneck in the Fast R-CNN process—the **region proposer**.
- The very first step to detecting the locations of objects is generating a bunch of potential bounding boxes or regions of interest to test.
- In Fast R-CNN, these proposals were created using **Selective Search**, a fairly **slow** process that was found to be the bottleneck of the overall process.
- In middle 2015, a team at Microsoft Research composed of Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, found a way to make the region proposal step almost cost free through an architecture they named Faster R-CNN.
- The insight of Faster R-CNN was that region proposals depended on features of the image that were **already calculated** with the forward pass of the CNN (first step of classification). **So why not reuse those same CNN results for region proposals instead of running a separate selective search algorithm?**

FASTER R-CNN: HOW IT WORKS

- Indeed, this is just what the Faster R-CNN team achieved.
- In their work, a single CNN is used to both carry out region proposals and classification. This way, **only one CNN needs to be trained** and we get region proposals almost for free! The authors write:
- *Our observation is that **the convolutional feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals** [thus enabling nearly cost-free region proposals*.
- Here are the inputs and outputs of their model:
  - **Inputs**: Images (Notice how region proposals are not needed).
  - **Outputs**: Classifications and bounding box coordinates of objects in the images.

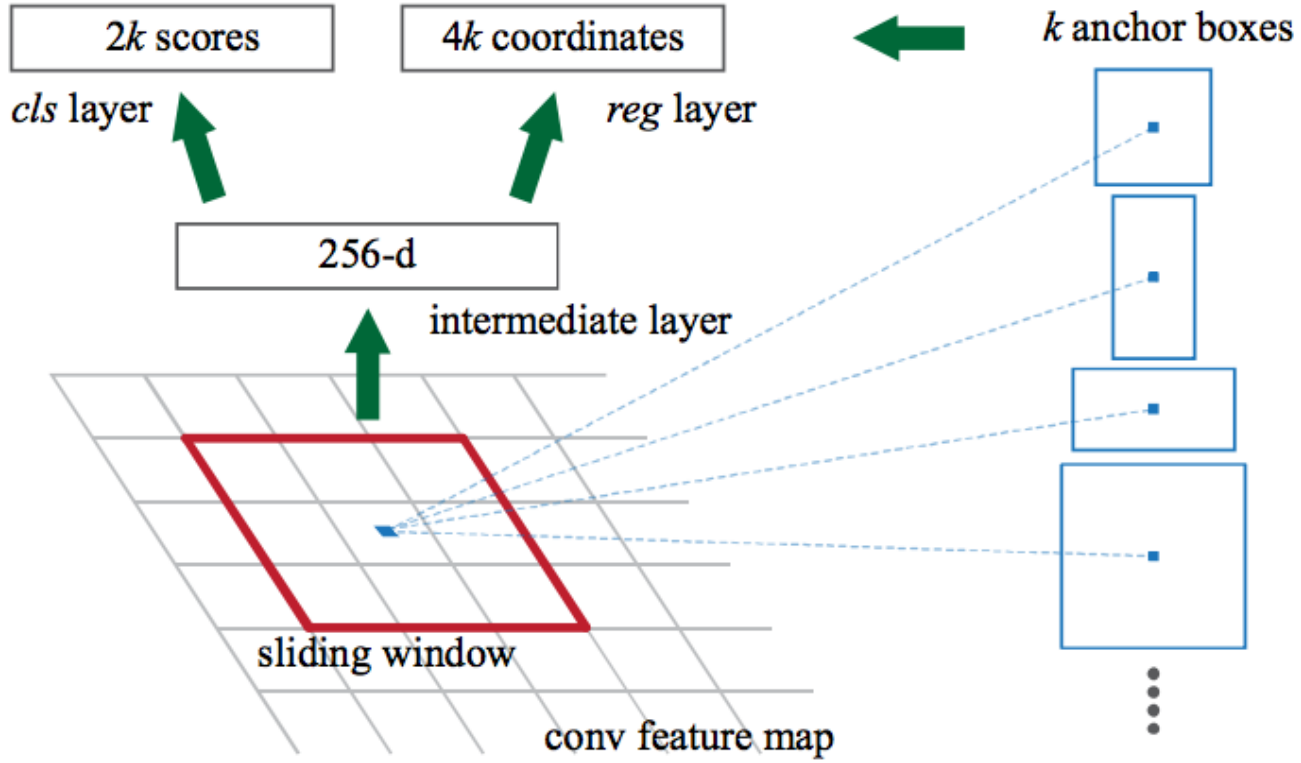
FASTER R-CNN: HOW THE REGIONS ARE GENERATED

- Let's take a moment to see how Faster R-CNN generates these region proposals from CNN features. Faster R-CNN adds a Fully Convolutional Network on top of the features of the CNN creating what's known as the **Region Proposal Network**.
- The Region Proposal Network works by passing a sliding window over the CNN feature map and at each window, outputting **k** potential bounding boxes and scores for how good each of those boxes is expected to be. What do these **k** boxes represent?



FASTER R-CNN: HOW THE REGIONS ARE GENERATED

- The Region Proposal Network slides a window over the features of the CNN. At each window location, the network outputs a score and a bounding box per anchor (hence 4k box coordinates where k is the number of anchors).

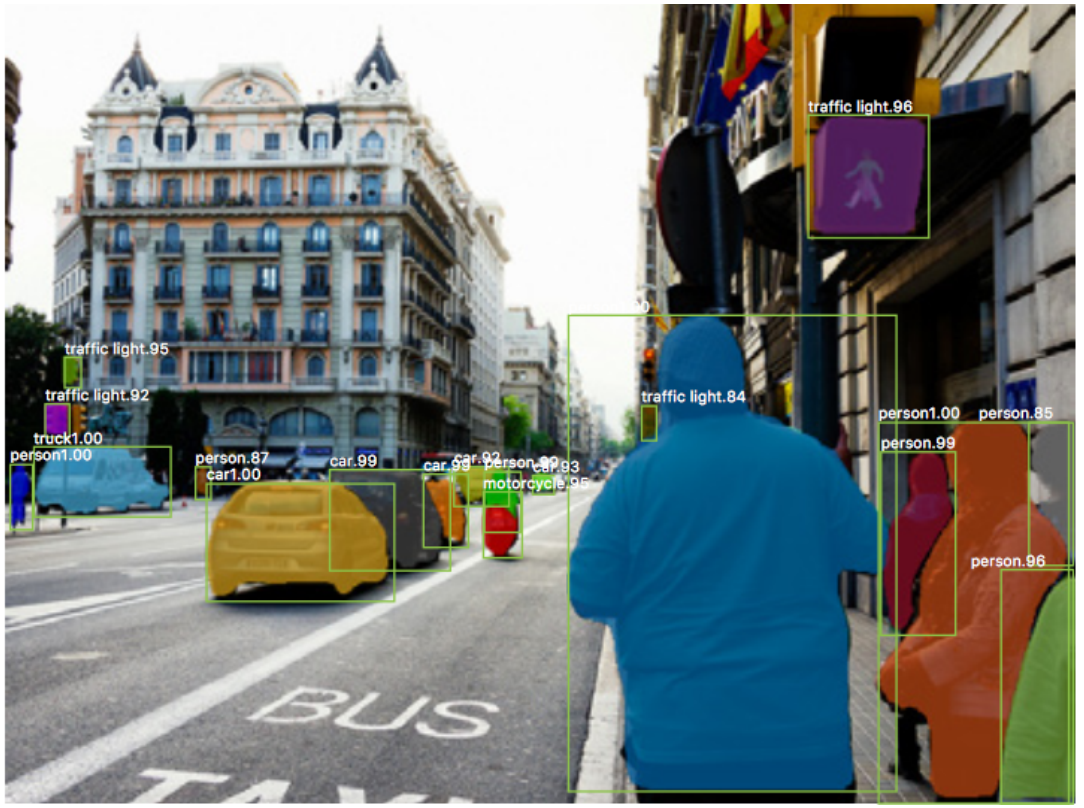


FASTER R-CNN: HOW IT WORKS

- Intuitively, we know that objects in an image should fit certain common aspect ratios and sizes. For instance, we know that we want some rectangular boxes that resemble the shapes of humans. Likewise, we know we won't see many boxes that are very very thin. In such a way, we create **k** such common aspect ratios we call **anchor boxes**. For each such anchor box, we output one bounding box and score per position in the image.
- With these anchor boxes in mind, let's take a look at the inputs and outputs to this Region Proposal Network:
- **Inputs**: CNN Feature Map.
- **Outputs**: A bounding box per anchor. A score representing how likely the image in that bounding box will be an object.
- We then pass each such bounding box that is likely to be an object into Fast R-CNN to generate a classification and tightened bounding boxes.

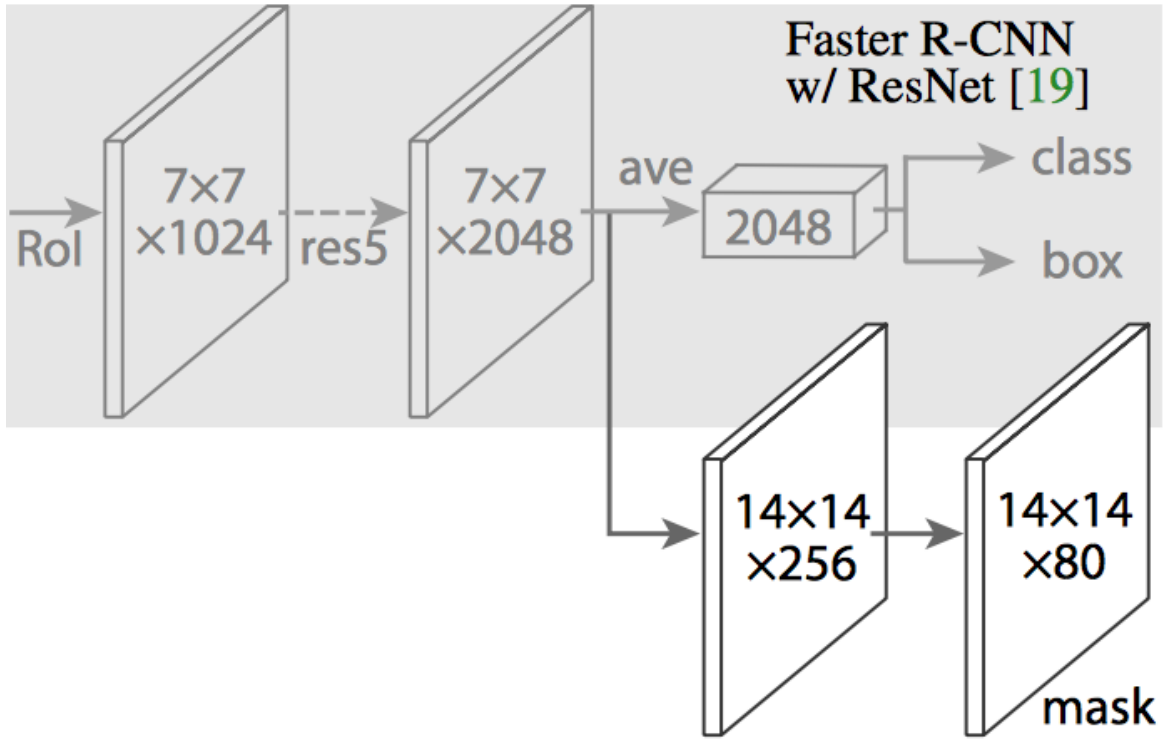
MASK R-CNN: EXTENDING FASTER R-CNN FOR PIXEL LEVEL SEGMENTATION (2017)

- So far, we've seen how we've been able to use CNN features in many interesting ways to effectively locate different objects in an image with bounding boxes.
- Can we extend such techniques to go one step further and locate exact pixels of each object instead of just bounding boxes?
- This problem, known as image segmentation, is what Kaiming He and a team of researchers, including Girshick, explored at Facebook AI using an architecture known as **Mask R-CNN**.
- Much like Fast R-CNN, and Faster R-CNN, Mask R-CNN's underlying intuition is straight forward.
- Given that Faster R-CNN works so well for object detection, could we extend it to also carry out pixel level segmentation?



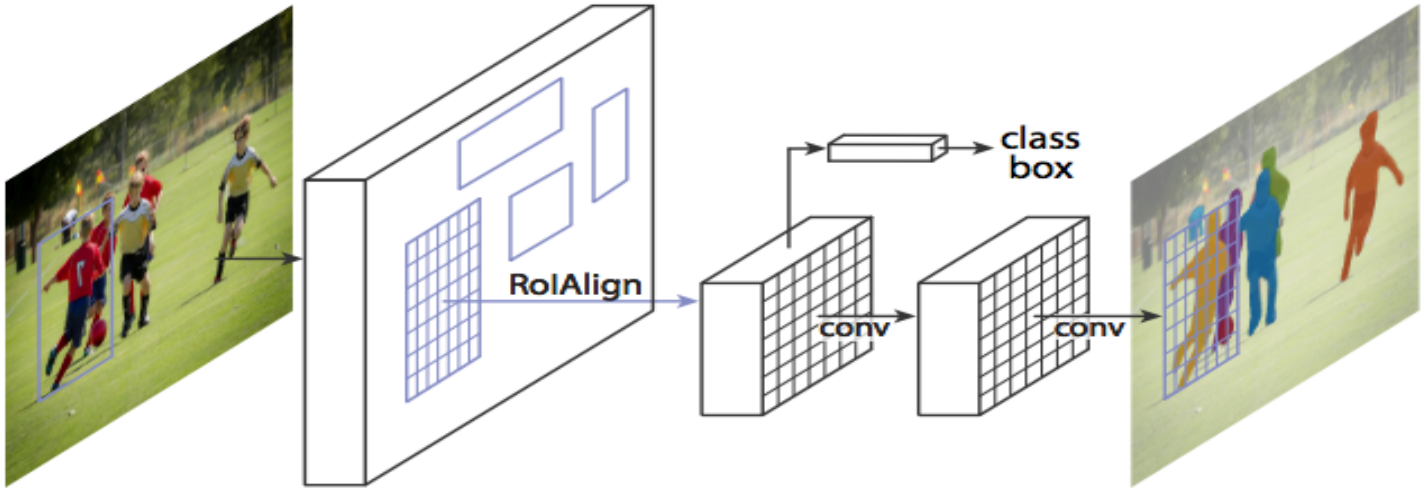
MASK R-CNN: HOW IT WORKS

- Mask R-CNN does this by adding a branch to Faster R-CNN that outputs a binary mask that says whether or not a given pixel is part of an object. The branch (in white in the above image), as before, is just a Fully Convolutional Network on top of a CNN based feature map. Here are its inputs and outputs:
- **Inputs**: CNN Feature Map.
- **Outputs**: Matrix with 1s on all locations where the pixel belongs to the object and 0s elsewhere (this is known as a binary mask).
- But the Mask R-CNN authors had to make one small adjustment to make this pipeline work as expected.



MASK R-CNN: ROI ALIGN

- When run without modifications on the original Faster R-CNN architecture, the Mask R-CNN authors realized that the regions of the feature map selected by RoIPool were slightly misaligned from the regions of the original image.
- Since image segmentation requires pixel level specificity, unlike bounding boxes, this naturally led to inaccuracies.
- The authors were able to solve this problem by cleverly adjusting RoIPool to be more precisely aligned using a method known as RoIAlign.
- Instead of RoIPool, the image gets passed through RoIAlign so that the regions of the feature map selected by RoIPool correspond more precisely to the regions of the original image. This is needed because pixel level segmentation requires more fine-grained alignment than bounding boxes.

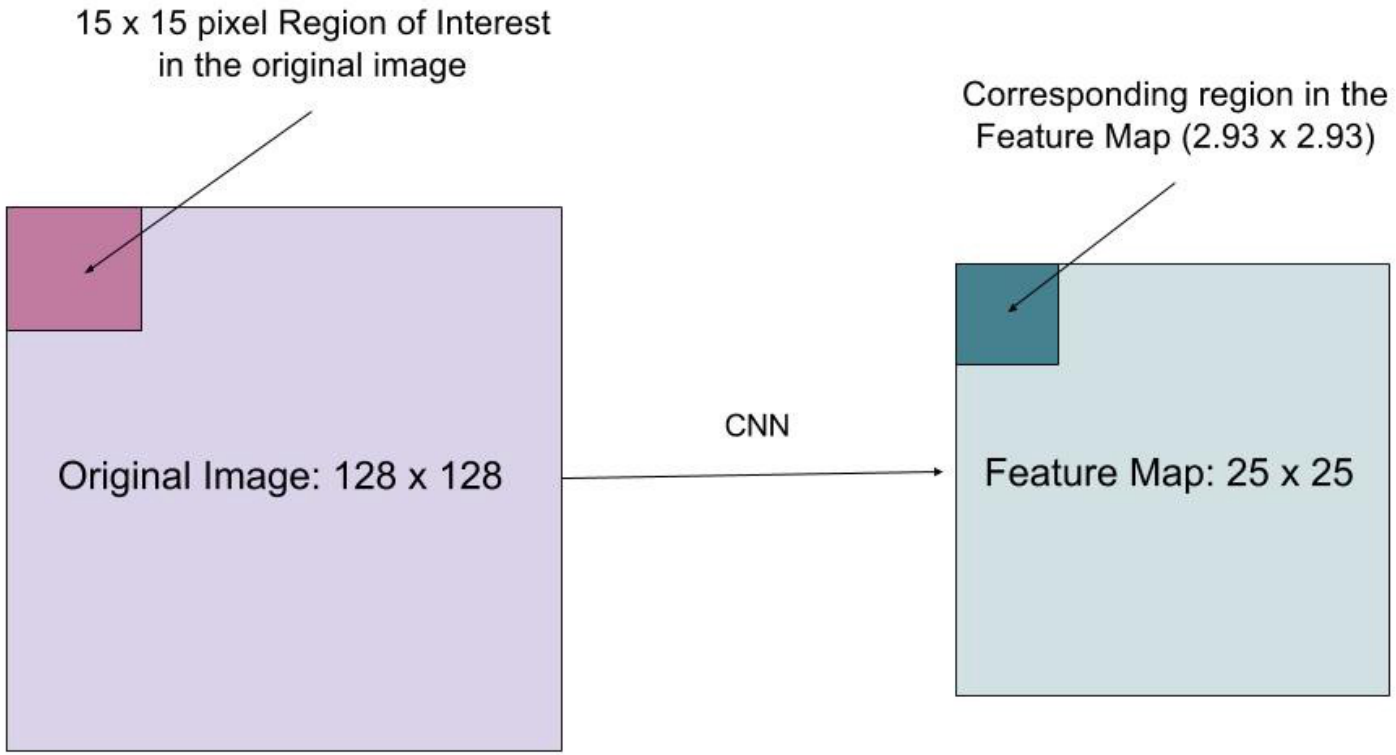


MASK R-CNN: ROI ALIGN

- Imagine we have an image of size **128x128** and a feature map of size **25x25**. Let's imagine we want features the region corresponding to the top-left **15x15**pixels in the original image (see above). How might we select these pixels from the feature map?
- We know each pixel in the original image corresponds to ~ 25/128 pixels in the feature map.
- To select 15 pixels from the original image, we just select 15 * 25/128 ~= **2.93** pixels.
- In RoIPool, we would round this down and select 2 pixels causing a slight misalignment.
- However, in RoIAlign, **we avoid such rounding.** Instead, we use bilinear interpolation to get a precise idea of what would be at pixel 2.93. This, at a high level, is what allows us to avoid the misalignments caused by RoIPool.

MASK R-CNN: ROI ALIGN

- How do we accurately map a region of interest from the original image onto the feature map?



MASK R-CNN

- Once these masks are generated, Mask R-CNN combines them with the classifications and bounding boxes from Faster R-CNN to generate such precise segmentations.
- So, Mask R-CNN is able to segment as well as classify the objects in an image.

TO SUM UP

- In just 3 years, we've seen how the research community has progressed from Krizhevsky et. al's original result on CNN to R-CNN, and finally all the way to such powerful results as Mask R-CNN.
- Seen in isolation, results like Mask R-CNN seem like incredible leaps of genius that would be unapproachable.
- However, advancements are really the sum of intuitive, incremental improvements through years of hard work and collaboration. Each of the ideas proposed by R-CNN, Fast R-CNN, Faster R-CNN, and finally Mask R-CNN were not necessarily quantum leaps, yet their sum products have led to really remarkable results that bring us closer to a human level understanding of sight.
- Let us see what the future holds!

ACKNOWLEDGMENTS

- The slides are based on:
- the blogpost by Dhruv Parthasarathy, "A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN", and
- Stanford's CS231N slides by Fei Fei Li, Andrei Karpathy, and Justin Johnson.
- Prof. N. Vassilas' slides on CNNs for postgraduate AI Course