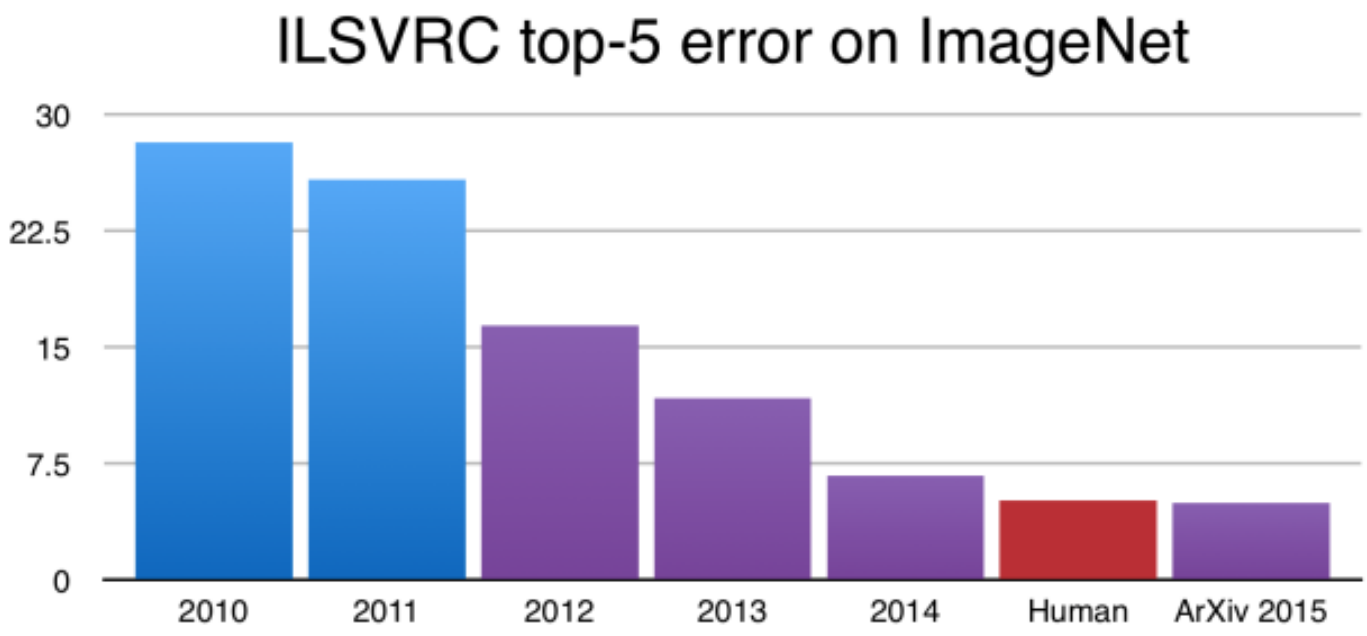
A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN

By Dhruv Parthasarathy | Apr. 22nd, 2017

At Athelas, we use Convolutional Neural Networks(CNNs) for a lot more than just classification! In this post, we'll see how CNNs can be used, with great results, in image instance segmentation.

Ever since Alex Krizhevsky, Geoff Hinton, and Ilya Sutskever won ImageNet in 2012, 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!



CNNs now outperform humans on the ImageNet challenge. The y-axis in the above graph is the error rate on ImageNet.

While these results are impressive, image classification is far simpler than the complexity and diversity of true human visual understanding.

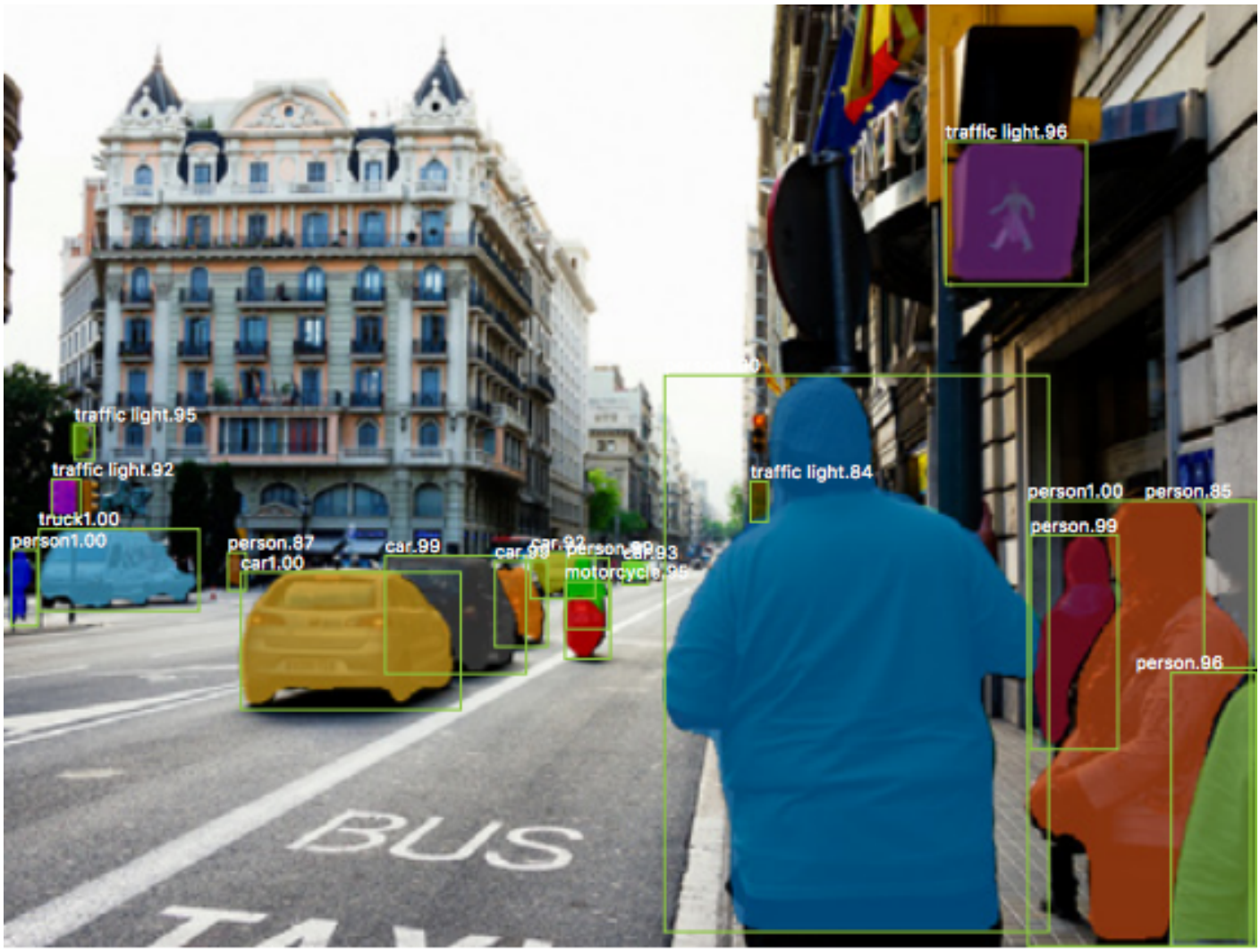
An example of an image used in the classification challenge. Note how the image is well framed and has just one object.

In classification, there's generally an image with a single object as the focus and the task is to say what that image is (see above). But when we look at the world around us, we carry out far more complex tasks.

Sights in real life are often composed of a multitude of different, overlapping objects, backgrounds, and actions.

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!



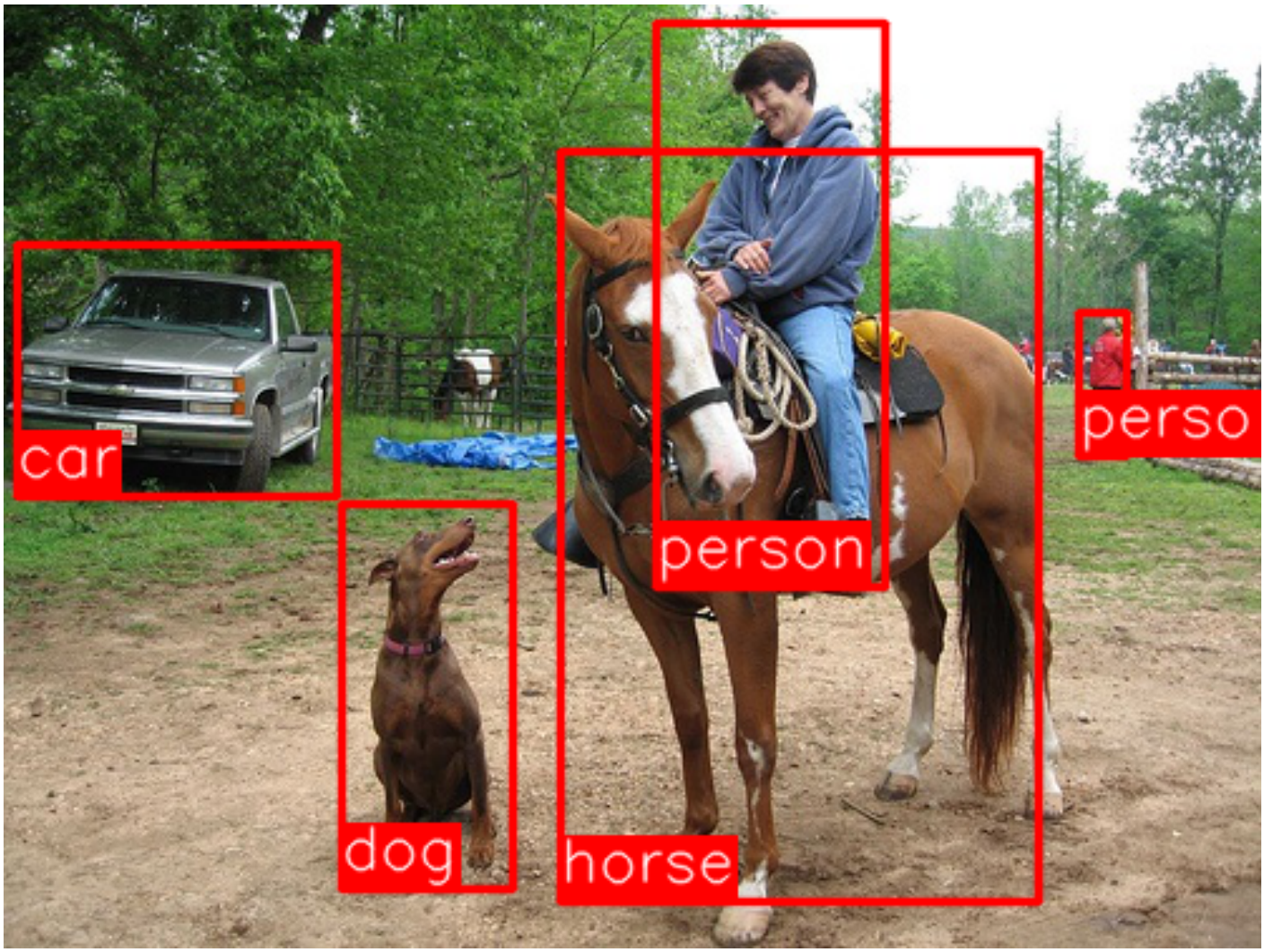
In image segmentation, our goal is to classify the different objects in the image, and identify their boundaries. Source: Mask R-CNN paper.

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? As has been shown by Ross Girshick and his peers over the last few years, the answer is conclusively yes.

Through this post, we'll cover the intuition behind some of the main techniques used in object detection and segmentation and see how they've evolved from one implementation to the next. In particular, we'll cover R-CNN (Regional CNN), the original application of CNNs to this problem, along with its descendants Fast R-CNN, and Faster R-CNN. Finally, we'll cover Mask R-CNN, a paper released

recently by Facebook Research that extends such object detection techniques to provide pixel level segmentation. Here are the papers referenced in this post:

1. R-CNN: https://arxiv.org/abs/1311.2524
2. Fast R-CNN: https://arxiv.org/abs/1504.08083
3. Faster R-CNN: https://arxiv.org/abs/1506.01497
4. Mask R-CNN: https://arxiv.org/abs/1703.06870



Object detection algorithms such as R-CNN take in an image and identify the locations and classifications of the main objects in the image. Source: https://arxiv.org/abs/1311.2524.

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 what today seems like an inevitable question:

> To what extent do [Krizhevsky et. al's results] generalize to object detection?

Object detection is the task of finding the different objects in an image and classifying them (as seen in the image above). The team, comprised of Ross Girshick (a name we'll see again), Jeff Donahue, and Trevor Darrel found that this problem can be solved with Krizhevsky's results by testing on the PASCAL VOC Challenge, a popular object detection challenge akin to ImageNet. They write,

> This paper is the first to show that a CNN can lead to dramatically higher object detection performance on PASCAL VOC as compared to systems based on simpler HOG-like features.

Let's now take a moment to understand how their architecture, Regions With CNNs (R-CNN) works.

Understanding R-CNN

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 does what we might intuitively do as well - **propose a bunch of boxes in the image and see if any of them actually correspond to an object.**

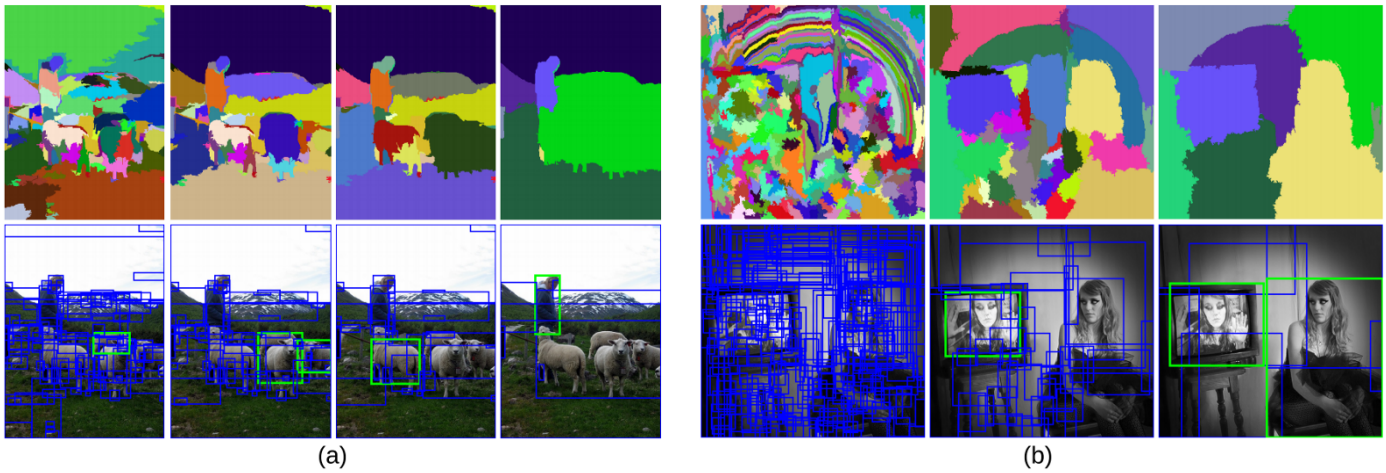
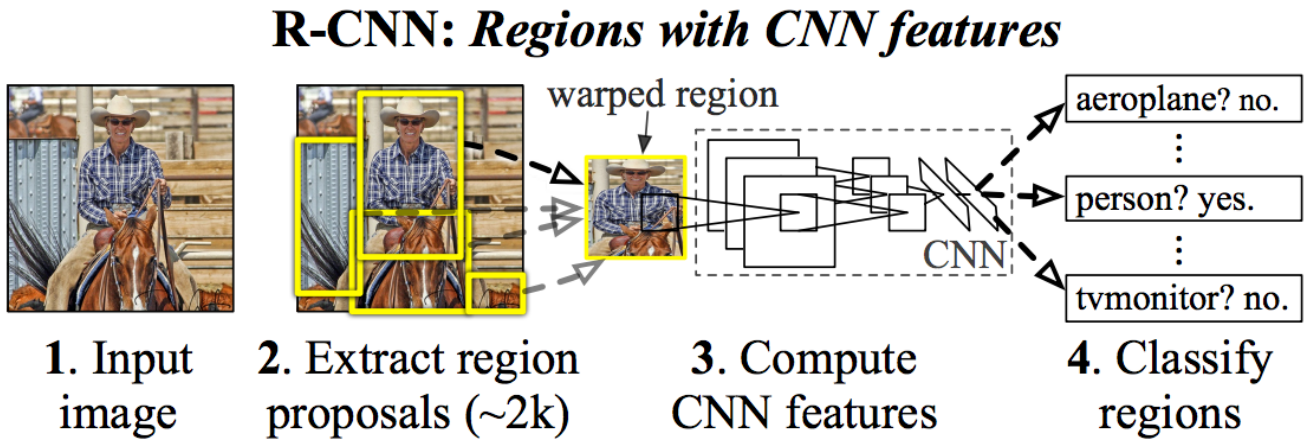


Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.

Selective Search looks through windows of multiple scales and looks for adjacent pixels that share textures, colors, or intensities. Image source: https://www.koen.me/research/pub/uijlings-ijcv2013-draft.pdf

R-CNN creates these bounding boxes, or region proposals, using a process called Selective Search which you can read about here. At a high level, Selective Search (shown in the image above) 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.



After creating a set of region proposals, R-CNN passes the image through a modified version of AlexNet to determine whether or not it is a valid region. Source: https://arxiv.org/abs/1311.2524.

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), as shown above.

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 in the image above.

Improving the Bounding Boxes

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:

1. Generate a set of proposals for bounding boxes.
2. 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.
3. Run the box through a linear regression model to output tighter coordinates for the box once the object has been classified.

Ross Girshick wrote both R-CNN and Fast R-CNN. He continues to push the boundaries of Computer Vision at Facebook Research.

R-CNN works really well, but is really quite slow for a few simple reasons:

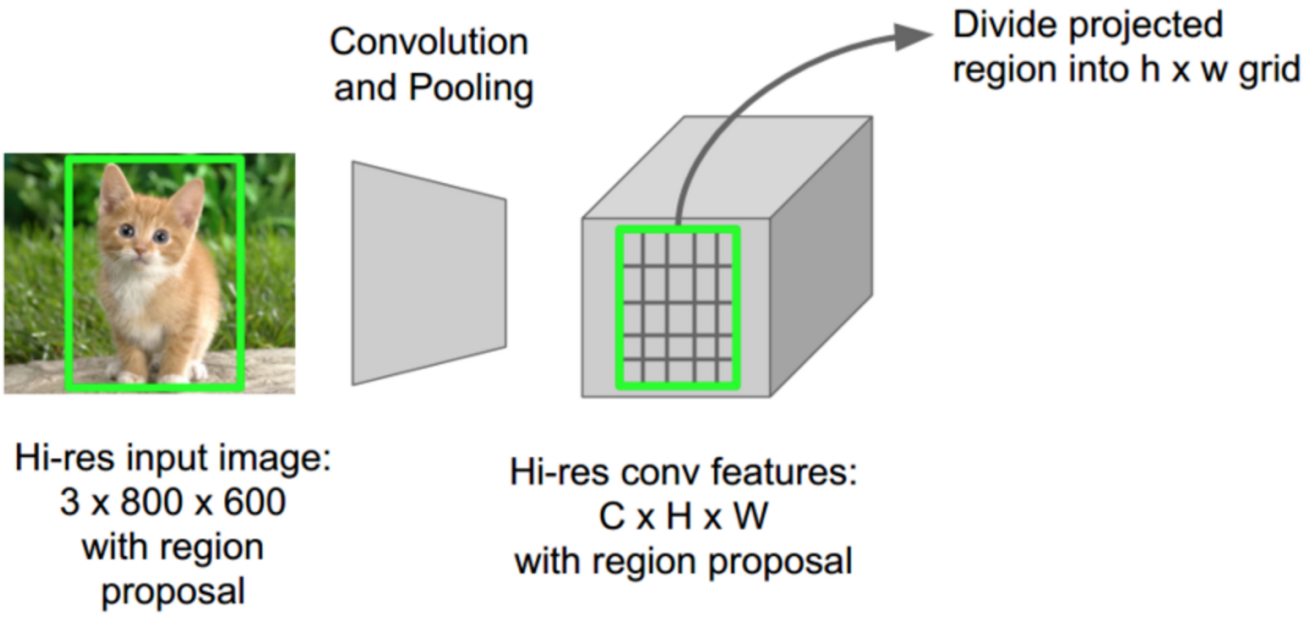
1. It requires a forward pass of the CNN (AlexNet) for every single region proposal for every single image (that's around 2000 forward passes per image!).
2. It has to train three different models separately - the CNN to generate image features, the classifier that predicts the class, and the regression model to

tighten the bounding boxes. This makes the pipeline extremely hard to train.

In 2015, Ross Girshick, the first author of R-CNN, solved both these problems, leading to the second algorithm in our short history - Fast R-CNN. Let's now go over its main insights.

Fast R-CNN Insight 1: RoI (Region of Interest) 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?**

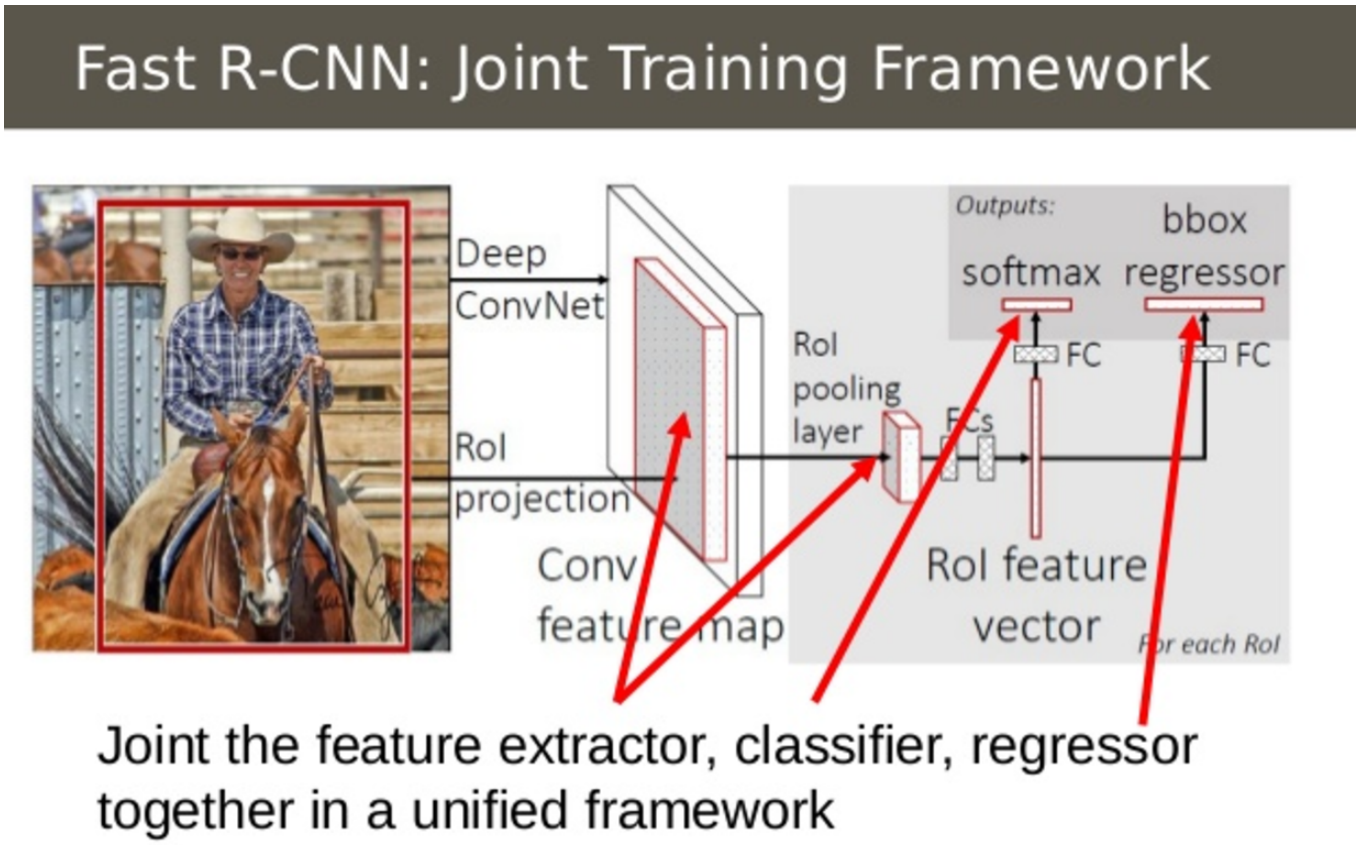


In RoIPool, a full forward pass of the image is created and the conv features for each region of interest are extracted from the resulting forward pass. Source: Stanford's CS231N slides by Fei Fei Li, Andrei Karpathy, and Justin Johnson.

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 image above, notice how 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 Insight 2: Combine All Models into One Network



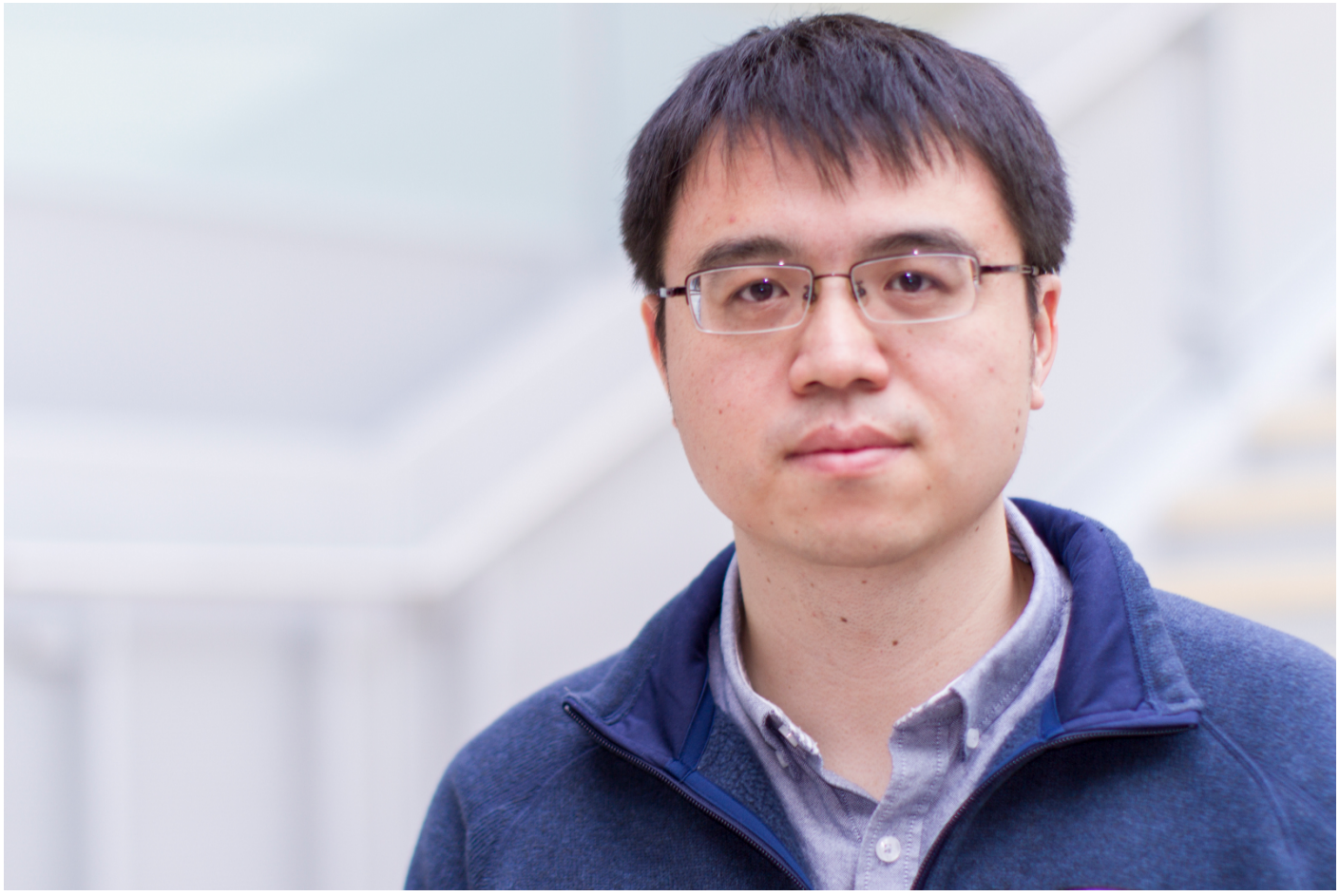
Fast R-CNN combined the CNN, classifier, and bounding box regressor into one, single network. Source: https://www.slideshare.net/simplyinsimple/detection-52781995.

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.**

You can see how this was done in the image above. 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.

Even with all these advancements, there was still one remaining bottleneck in the Fast R-CNN process—the region proposer. As we saw, 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.



Jian Sun, a principal researcher at Microsoft Research, led the team behind Faster R-CNN. Source: https://blogs.microsoft.com/next/2015/12/10/microsoft-researchers-win-imagenet-computer-vision-challenge/#sm.00017fqnl1bz6fqf11amuo0d9ttdp

In the 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 (creatively) 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?**

In Faster R-CNN, a single CNN is used for region proposals, and classifications. Source: https://arxiv.org/abs/1506.01497.

Indeed, this is just what the Faster R-CNN team achieved. In the image above, you can see how 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.

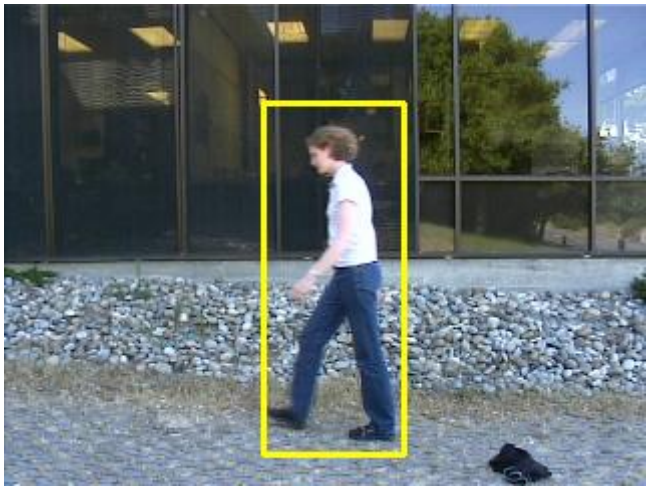
How the Regions are Generated

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 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). Source: https://arxiv.org/abs/1506.01497.

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?



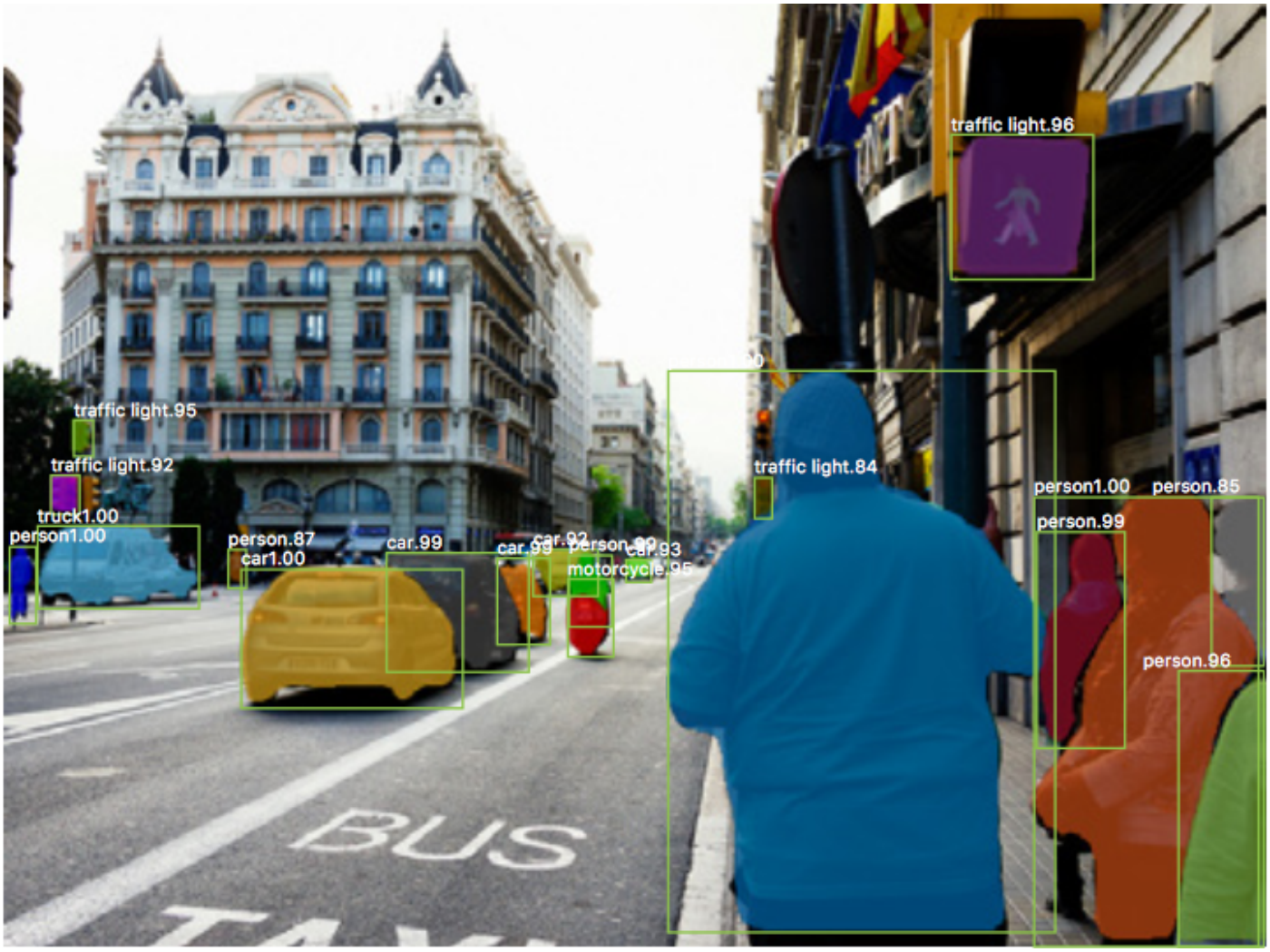
We know that the bounding boxes for people tend to be rectangular and vertical. We can use this intuition to guide our Region Proposal networks through creating an anchor of such dimensions. Image Source: http://vlm1.uta.edu/~athitsos/courses/cse6367_spring2011/assignments/assignment1/bbox0062.jpg.

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.

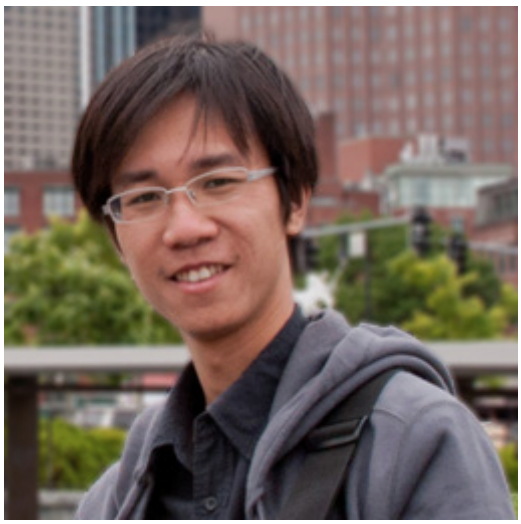


The goal of image instance segmentation is to identify, at a pixel level, what the different objets in a scene are. Source: https://arxiv.org/abs/1703.06870.

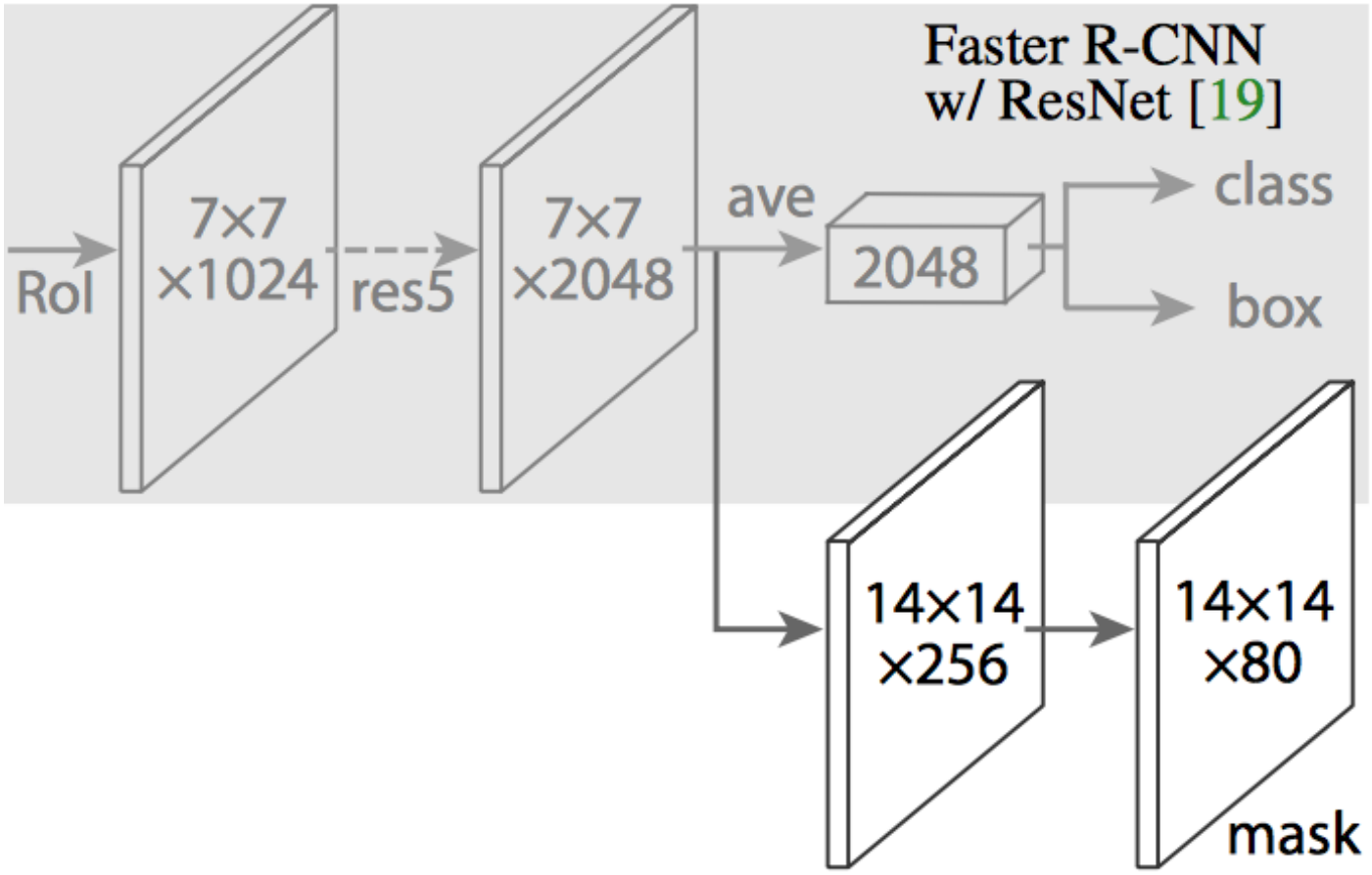
So far, we've seen how we've been able to use CNN features in many interesting ways to effectively locate different objects in an image with bounding boxes.

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?



Kaiming He, a researcher at Facebook AI, is lead author of Mask R-CNN and also a coauthor of Faster R-CNN.



In Mask R-CNN, a Fully Convolutional Network (FCN) is added on top of the CNN features of Faster R-CNN to generate a mask (segmentation output). Notice how this is in parallel to the classification and bounding box regression network of Faster R-CNN. Source: https://arxiv.org/abs/1703.06870.

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.

RoiAlign - Realigning RoIPool to be More Accurate

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. Source: https://arxiv.org/abs/1703.06870.

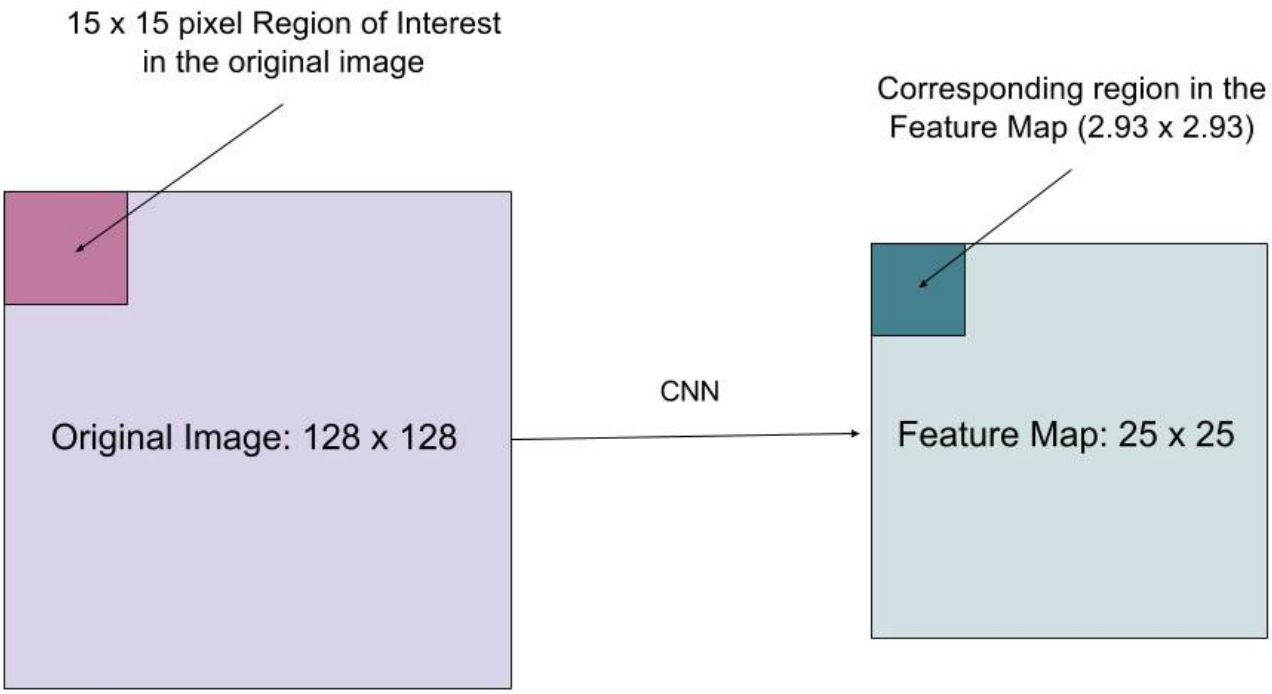
When run without modifications on the original Faster R-CNN architecture, the Mask R-CNN authors realized that the regions of the feature map selected by RoIPool were slightly misaligned from the regions of the original image. Since image segmentation requires pixel level specificity, unlike bounding 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.

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



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

a high level, is what allows us to avoid the misalignments caused by RoIPool.

Once these masks are generated, Mask R-CNN combines them with the classifications and bounding boxes from Faster R-CNN to generate such wonderfully precise segmentations:

If you're interested in trying out these algorithms yourselves, here are relevant repositories:

Faster R-CNN

Mask R-CNN

In just 3 years, we've seen how the research community has progressed from Krizhevsky et. al's original result 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. Yet, through this post, I hope you've seen how such 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.

What particularly excites me, is that the time between R-CNN and Mask R-CNN was just three years! With continued funding, focus, and support, how much further can Computer Vision improve over the next three years?

Dhruv Parthasarathy you can try **SSD Tensorflow** very easily (especially if you use my **gtarobotics/udacity-sdc** Docker image), see the IPython script here, I created to test SSD on driving videos (even streamed directly from YouTube), for self driving cars (SDC) in **OSSDC.org**:

> https://github.com/OSSDC/SSD-Tensorflow/...

Hi Dhruv:

I am in the DL ND. Is there any example in github and some datasets that I can use it to learn the technique. It looks very very interesting to me.

Thanks

Suman

Hi Suman!

The code for Faster R-CNN can all be found here thanks to Girshick: https://github.com/rbgirshick/py-faster-rcnn.

Good luck with the program!

Dhruv

Nice article. Mask R-CNN definitely looks like a good candidate for blood diagnostics. I personally like SSD because its real time and can be applied with squeezenet type of models. Good luck and congrats on doing this cool work :D

Thanks! I haven't tried out SSD but Marius Slavescu also speaks highly of it so I'll check it out! Good luck with your work as well Vivek!

Dhruv

Dhruv Parthasarathy Very clear writing, thanks for the post!

Quick question: in the graphic for the Regional Proposal Network (Faster-RCNN), it says 2k scores for the classification layer. Why is this 2k and not 1k i.e. why two scores per anchor?

Hi Rama!

It's 2k because the network as shown in the graph outputs 2 scores per image:

- How likely it is to be an object
- How likely it is to **not** be an object

You can also do this with just one score as the other is just 1 minus the first score. Either way works!

There are quite a few other techniques which I think are more promising than R-CNN, e.g. recently published (few weeks ago) "Accurate Single Stage Detector Using Recurrent Rolling Convolution" https://arxiv.org/abs/1704.05776 or Single Shot approaches such as SSD or YOLO (which have better trade-offs). In my opinion research in Detection is now...

Hi Akshay!

You’re probably right that there are better techniques than R-CNN. My goal was to trace the creation of Mask R-CNN and it naturally led all the way back to R-CNN as it was one of the first attempts at applying CNNs to PASCAL VOC (from what I know).

Thanks for your feedback!

Dhruv

Brilliant summary and explanations of these recent techniques !

Thanks for the kind words Julien - If there’s anything I can improve let me know!

---

https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4